NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

USING OBJECT-ORIENTED DATABASES FOR IMPLEMENTATION OF INTERACTIVE ELECTRONIC TECHNICAL MANUALS

by

Evyatar Chelouche

March, 1992

Thesis Advisor: C. Thomas Wu

Approved for public release; distribution is unlimited.

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: Computer Science Dept. Naval Postgraduate School; 6b. OFFICE SYMBOL (if applicable): 37	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
Using Object-Oriented Databases for Implementation of Interactive Electronic Technical Manuals

12. PERSONAL AUTHOR(S)
Evyatar Chelouche

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Progress	FROM 10/90 TO 03/92	March 1992	102

16. SUPPLEMENTARY NOTATION
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the United States Government

17. COSATI CODES		
FIELD	GROUP	SUB-GROUP

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Object-Oriented Databases, Interactive Electronic Technical Manual, Computer-aided Acquisition and Logistic Support.

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

Computer-aided Acquisition and Logistic Support (CALS) is a Department of Defense (DoD) and Industry strategy to transition from paper-intensive acquisition and logistic processes to a highly automated and integrated mode of weapon system acquisition and operation. A newly demonstrated technology in the context of the CALS initiative is the Interactive Electronic Technical Manual (IETM), which is a portable computer system developed for the use of technicians maintaining weapon systems. The introduction of IETM systems will relieve the technician from the need to carry extensive volumes of hard-copy technical manuals, provide him with easy interactive access to the required technical data and is expected to have a profound impact on the way weapon systems maintenance is conducted and the costs associated with it.

Object-Oriented Database Management Systems (OODBMS) is a new technology that marries the characteristics of object-oriented programming languages and data persistence provided by database systems. This thesis explores issues related to the utilization of OODBMS for the implementation of IETM databases, discusses the benefits of this approach and addresses some architectural issues of OODBMS in the context of IETM implementation.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: [X] UNCLASSIFIED UNLIMITED [ ] SAME AS RPT. [ ] DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: C. T. Wu	22b. TELEPHONE (Include Area Code): (408) 646-3391; 22c. OFFICE SYMBOL: 52/Wq

Approved for public release; distribution is unlimited.

Using Object-Oriented Databases
for Implementation of
Interactive Electronic Technical Manuals

by

Evyatar Chelouche
Major, Israeli Air Force
B.S., Technion, Israel Institute of Technology, 1985

Submitted in partial fulfillment
of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 1992

Author: _____
Evyatar Chelouche

Approved by: _____
C. T. Wu, Thesis Advisor

_____
CDR B. B. Ciannotti, Second Reader

_____
Robert B. McGhee, Chairman
Department of Computer Science

ABSTRACT

Computer-aided Acquisition and Logistic Support (CALS) is a Department of Defense (DoD) and Industry strategy to transition from paper-intensive acquisition and logistic processes to a highly automated and integrated mode of weapon system acquisition and operation. A newly demonstrated technology in the context of the CALS initiative is the Interactive Electronic Technical Manual (IETM), which is a portable computer system developed for the use of technicians maintaining weapon systems. The introduction of IETM systems will relieve the technician from the need to carry extensive volumes of hard-copy technical manuals, provide him with easy interactive access to the required technical data and is expected to have a profound impact on the way weapon systems maintenance is conducted and the costs associated with it.

Object-Oriented Database Management Systems (OODBMS) is a new technology that marries the characteristics of object-oriented programming languages and data persistence provided by database systems. This thesis explores issues related to the utilization of OODBMS for the implementation of IETM databases, discusses the benefits of this approach and addresses some architectural issues of OODBMS in the context of IETM implementation.

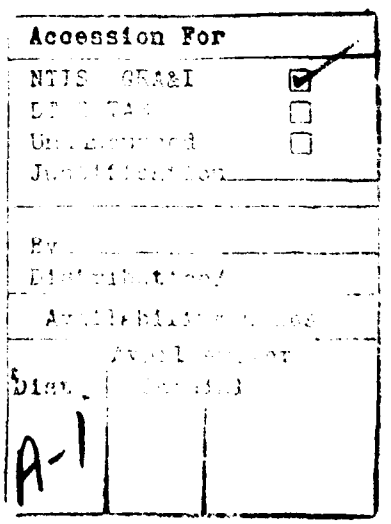

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CAD	Computer Aided Design
CAE	Computer Aided Engineering
CALS	Computer-aided Acquisition and Logistic Support
CAM	Computer Aided Manufacturing
CDM	Content Data Model
CIM	Computer Integrated Manufacturing
DBMS	Data Base Management System
DOD	Department of Defense
DON	Department of the Navy
DTD	Data Type Definition
EDS	Electronic Display System
IETM	Interactive Electronic Technical Manual
IETMDB	Interactive Electronic Technical Manual Data Base
ILS	Integrated Logistic Support
LSA	Logistic Support Analysis
LSAR	Logistic Support Analysis Record
NSIA	National Security Industrial Association
OODBMS	Object-Oriented Data Base Management System
OOP	Object-Oriented Paradigm

OOPL	Object-Oriented Programming Languages
OSD	Office of the Secretary of Defense
PDES	Product Data Exchange Standard
RDBMS	Relational Data Base Management System
SGML	Standard Generalized Markup Language
TID	Technical Information Document

ACKNOWLEDGEMENTS

I would like to thank the following individuals for their assistance in preparation of this thesis:

1. CDR B. B. Giannotti, for his invaluable help in establishing contacts within NAVAIR, and his assistance in creating the framework for my research in the CALS domain.

2. Mr. Shawn P. Magill (AIR-41144), and members of his staff, especially Ms. Cynthia Janosky, for organizing a very educational tour to NAVAIR and DTRC, and for continuous support in obtaining background material on the implementation of CALS in the Navy.

3. Mr. Joseph Garner and other staff members of the David Taylor Research Center, CALS Technology Integration Lab, for the details on their current research in IETM systems and IETM specifications.

4. CDR Stephen M. Carr (AIR-41142C), on enlightening discussions which assisted me in narrowing down the vast research domain that exists in CALS to the one topic which I selected as the focus of this thesis.

5. Prof. C. Thomas Wu, my thesis advisor, for his guidance throughout this thesis effort and for facilitating my attendance at the 1991 CALS Exposition and Conference. Both experiences were invaluable to me in the process of pursuing expertise in CALS systems.

I. INTRODUCTION

Computer-aided Acquisition and Logistic Support (CALS) is a Department of Defense (DoD) and Industry strategy to transition from paper-intensive acquisition and logistic processes to a highly automated and integrated mode of weapon systems acquisition and operation. The CALS initiative has set three major goals for changing the existing relationship between government and industry: reduction of procurement lead time by creating integrated and shared databases between government and industry, reduction of weapon systems life-cycle costs by transitioning from a paper intensive mode of information exchange to a digital information exchange based on accepted standards, and improving system quality by providing integrated databases for design, manufacturing and logistics.

From the user perspective, CALS signals a new era in development of computerized systems based on new hardware and software. One of the newly demonstrated technologies in the context of the CALS initiative is the Interactive Electronic Technical Manual (IETM), which is expected to have a profound impact on the way weapon systems maintenance is conducted and the costs associated with it.

The requirement for creation of integrated and shared databases as part of the future CALS system environment, provides the motivation for evaluation of new database technology. The current focus of database technology research is on Object-Oriented Database Management Systems (OODBMS), a technology that marries the characteristics

of object-oriented programming languages and data persistence provided by database systems.

The purpose of this thesis is to pursue new knowledge in a domain which is at the intersection of the front edge of database technology research and the new technologies evolving in the context of the CALS initiative. More specifically, this thesis will explore issues related to the utilization of object-oriented database technology for the implementation of Interactive Electronic Technical Manual (IETM) systems.

The thesis is structured as follows: Chapter II contains the problem statement and an outline of the thesis goals. Chapter III reviews the evolution of database technology, and provides background information on the characteristics of object-oriented languages and object-oriented databases as well as recent attempts to develop a new methodology that combines the relational and object-oriented approaches. Chapters IV through VI provide the background for understanding the systems domain of CALS: Chapter IV contains a broad overview of the CALS initiative. Chapter V narrows the CALS perspective to the domain of CALS systems in the Navy, whereas Chapter VI focuses on a specific CALS system, namely the IETM and the IETM Data Base (IETMDB). Chapter VII presents the proposed solution of utilizing OODB technology for the implementation of IETMDB. Chapter VIII discusses the benefits of this approach by comparing it to the alternative approach of implementing an IETMDB with relational database technology. It also discusses some architectural issues of object-oriented databases in the context of IETMDB implementation. Concluding remarks are in Chapter IX, followed by notes for future research in Chapter X.

II. PROBLEM STATEMENT

A. THESIS FOCUS

CALS provides new horizons for both the researcher in computer science and the developer of computer applications. Many unexplored issues need to be addressed before CALS can be fully implemented in existing or future systems environments. Some areas that require further research are within the domains of information management, database architecture and implementation, telecommunications, interconnection of computerized systems, definition of standards and development of software engineering tools.

This thesis will focus on the domain of database management systems. The primary focus will be on the application of OODBMS for the implementation of IETMDB systems. However, to provide a better understanding of the capabilities provided by the object-oriented approach, a secondary focus will be placed on the study of its capabilities in comparison to those provided by the well established Relational Data Base Management System (RDBMS) approach. This secondary focus will be in the context of IETMDB systems as well.

B. THESIS GOALS

The following are the goals of this thesis:

1. Provide better understanding as to the potential use of OODBMS in the implementation of IETMDB.

2. Provide better understanding of the advantages and disadvantages in using different database methodologies for IETMDB implementation.

3. Identify potential enhancements to the IETMDP concept through application of advanced concepts of database architecture.

C. OPEN QUESTIONS

A finer definition of the thesis goals is contained in the following set of open research questions:

1. How can object-oriented features such as encapsulation, inheritance and polymorphism be used in the implementation of an IETMDB?

2. What are the advantages/disadvantages of implementing an IETMDB using an OODBMS vs. using a RDBMS?

3. What form of database architecture is needed to support a DBMS with both relational and object-oriented capabilities, such that existing relational databases can be linked together with future object-oriented databases, for the purpose of IETMDB implementation?

4. How can object-oriented technology assist in enhancing the IETM concept to include additional functional capabilities in the logistics and maintenance domain?

The answers to these questions were sought throughout the entire thesis research effort. The details and findings are provided in the following chapters.

III. DATABASE TECHNOLOGY REVIEW

A. EVOLUTION OF DATABASE TECHNOLOGY

Database technology has undergone four generations of evolution, beginning with file systems, hierarchical database systems, network database systems and relational database systems. Although the hierarchical, network and relational data models have the same modeling power (Ullman, 1982), Relational Data Base Management Systems (RDBMS) are by far the most popular DBMS's. This is mainly due to the firm theoretical foundation on which they stand, as well as the ease of their design, maintenance and use (Bracket, 1987).

Relational database technology has limitations too. Amongst these are the inability to define compound (complex) entities, a limited set of data types, and the inability to define different versions of data (versioning). This imposes a difficulty when creating a data model for applications in the areas of engineering, manufacturing and multimedia systems. These applications require advanced facilities for modeling complex nested entities (e.g. engineering objects and compound documents), user-defined data types and long unstructured data (images, audio, etc.) and versioning of data (Kim, 1991, p. 21). These modeling facilities have become available by means of the object-oriented paradigm.

B. OBJECT-ORIENTED PROGRAMMING LANGUAGES

Object-oriented Programming Languages (OOPL) were developed as tools for realizing the object-oriented paradigm. This paradigm introduced new semantic concepts for data modeling: objects and classes. Objects are an encapsulation of data (attributes or instance variables) and behavior (methods). Instances of objects that have the same set of attributes and share the same methods are grouped in classes. Classes are organized in a class hierarchy in which classes inherit attributes and behavior from direct and indirect ancestor classes (Kim, 1991, pp. 21-22). The domain of an attribute can be any class, thus facilitating definition of complex objects which can have simple or complex objects as their attributes values (Kim, 1991, pp. 21-22).

Object-oriented design, and the use of OOPL, have improved code reusability, maintainability and modularity (Cox, 1986). However OOPL, as in other programming languages, lack the capability of supporting persistent data, i.e. data that can survive beyond a single programming session (Andrews and Harris, 1987, p. 430). Databases offer this capability which is the basis for extending the object-oriented paradigm to the database domain.

C. OBJECT-ORIENTED DATABASES

Object Oriented Data Base Management Systems (OODBMS) are emerging as the new generation of DBMS technology, and come as a natural extension to OOPL. OODBMS offer both the advanced data modeling concepts (encapsulation, inheritance,

aggregation) that were unavailable in earlier DBMS technologies, and support of persistent objects, which the OOPL are lacking.

Although OODBMS technology has great potential, it currently suffers from several weaknesses: First, the weakness of a data model exists due to the absence of a standard, and the lack of a formal foundation for a database language. Second, the complexity of the data model and database language, create difficulties for implementors, as well as users (Kim, 1991, pp. 26-29) (Premerlani et al., 1990, p. 99). This state of relative immaturity has been the grounds on which intermediate solutions have been constructed, which combine relational and object-oriented concepts.

D. COMBINING RELATIONAL AND OBJECT-ORIENTED CONCEPTS

Several approaches exist for integrating relational concepts and object-oriented concepts in a single DBMS. One approach is using an object-oriented interface layer that hides the details of the RDBMS from the application. The programmer uses an OOPL for his application, but the interface translates the operations to RDBMS service requests (Premerlani et al., 1990, pp. 99-109).

Another approach is to add relational capabilities to an existing OODBMS, thus enabling continued use of the well known relational model in a OODBMS environment. This allows conventional relational applications to include complex data (objects) (Nelson et al., 1990)(NPS52-90-025, May 1990, p. 3).

Taking into consideration other approaches that advocate maintaining separation of RDBMS and OODBMS (for reasons of performance optimization, technology availability

etc.), one can see the difficulty in developing a strategy for implementing a DBMS. This thesis will compare various methodologies for implementing a DBMS for the IETM system, which is one of the promising innovations in the domain of CALS systems. An overview of the CALS program is the subject of the following chapter.

IV. COMPUTER-AIDED ACQUISITION AND LOGISTIC SUPPORT

A. BACKGROUND

The advance of computer technology throughout recent years created the basis for increased automation of design and manufacturing processes, thereby introducing new systems for Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), Computer Aided Engineering (CAE), and Computer Integrated Manufacturing (CIM). As complexity of the design, production, maintenance and use of various industrial products increased, so did the volume of information created during the different stages of the product life-cycle. The organizations procuring these products were faced with an ongrowing influx of data and information, thus complicating the acquisition and system support tasks, resulting in an increase of systems *life-cycle costs* and an overall reduction of system reliability, maintainability, and availability.

The Department of Defense (DoD), as one of the main U.S. Government agencies involved in system procurement, initiated the Computer-aided Acquisition and Logistic Support (CALS) drive in 1985, to control information flow between Industry and DoD agencies. CALS is a joint DoD and Industry strategy aimed at automating the paper-intensive acquisition and logistic processes, facilitating data integration, exchange and access between government and industry maintained databases, thus eliminating duplicate data, and providing a framework for integrating existing islands of automation within DoD and Industry. CALS focuses on generation, access, management, maintenance,

distribution and use of technical data associated with weapon systems, such as engineering drawings, product definition and logistic support analysis data, technical and training manuals, etc.(DoD-OSD, CALS, 1989, pp. 3-5).

B. CALS OBJECTIVES

Three main objectives were defined for the CALS program (DoD-OSD, CALS, 1989, p. 9):

1. Reduce lead time. A shared and integrated data environment is expected to contribute to the shortening of weapon system design, development, production, maintenance and resupply activities.

2. Reduce life-cycle costs. A transition from a paper-intensive mode of information exchange to a digital technical information exchange based on accepted standards is expected to increase savings, as well as reduce the extent of duplicate data.

3. Improve system quality. Integrated databases will improve design and manufacturing by providing the basis for integrating reliability and maintainability factors into CAD/CAE tools and enhancing data consistency.

Digital file exchanges are the goal for the near term, between now and the mid 1990s, whereas advanced technologies and shared and integrated product databases are the goal set for the mid 1990s and beyond (DoD-OSD, CALS, 1989, p. 5). To meet these goals, DoD has defined both policy (DoD Directive 5000.2, Part 6, Section N) and implementation strategy to be carried out by the different services (DoD-OSD, CALS, 1989, pp. 11-19).

C. CALS IMPLEMENTATION STRATEGY

Five areas have been identified as being critical to the implementation of CALS: development of data-interchange standards, development and demonstration of new technology, acquisition policy and program management guidance, DoD infrastructure modernization and training to affect the necessary cultural change (DoD-OSD, CALS, 1989, pp. 11-19) (NSIA, "CALS: Making it happen", 1991, pp. 4-5). Three of these areas are related to the topic of this thesis:

1. CALS Standards

Transitioning from a paper-intensive mode of information exchange to a digital file transfer and distributed database access mode, requires definition of a common interface between DoD and Industry. For this purpose several functional, technical and data-management standards have been defined.[1]

2. Technology Development and Demonstration

Research and development (R&D) are carried out in several areas (DoD-OSD, CALS, 1989, p. 15):

---

[1] Initial standards published in the context of the CALS initiative were:
MIL-STD-1840A *Automated Interchange of Technical Information*,
MIL-D-28000 *Initial Graphics Interchange Standard (IGES)*,
MIL-M-28001 *Standard Generalized Markup Language (SGML)*,
MIL-R-28002 *Standard for Raster Images Representation*,
MIL-D-28003 *Computer Graphics Metafile (CGM) Representation*,
MIL-STD-1388-2B *Requirements for Logistic Support Analysis Record (LSAR)*.
The standardization process is an ongoing process and additional standards can be expected in the future. Current efforts are targeted on the definition of a Product Data Exchange Standard (PDES).

1. Development of a Product data Exchange Standard (PDES) to support a neutral representation of product data between dissimilar computer systems.

2. Integration of Reliability and Maintainability (R&M) analysis into the design process.

3. Logistic applications in the domain of electronic technical manuals, portable delivery devices, ordering of parts, etc.

4. Database management and access in the distributed system environment of DoD and Industry.

5. Application of CALS technologies in leading weapon system development programs.

From the above mentioned R&D areas, this thesis will explore issues related to both the development of logistic applications and database management.

3. DoD Infrastructure Modernization

The DoD has identified a need to modernize its system infrastructure to support the reception , integration, acces and use of digital technical information. This process encompasses systems for electronic technical manuals, engineering drawing repositories, computer-aided design and technical information integration (DoD-OSD, CALS, 1989, p. 19). For practical reasons, the scope of this thesis will be narrowed down to the systems domain defined in the Department of the Navy (DoN) CALS Architecture/Implementation Plan.

D. DON CALS ARCHITECTURE/IMPLEMENTATION PLAN

Guidelines and requirements for implementation of the CALS program within the Navy were initially defined in the DoN, CALS Strategic Plan, 1988 (pp. 1-7), and were

later expanded and detailed in the DoN, CALS Architecture/Implementation Plan, 1991 (pp. 1-30). The plan defined three phases for the implementation of CALS, as well as the application development and infrastructure modernization that will take place during each phase:

1. Implementation Plan Phases

1. Phase I (1988 - 1991). Use of first generation CALS standards in weapon system development and definition of an architecture to support receipt, storage, handling and use of digital technical information.

2. Phase II (1992 - 1996). Application of second generation CALS standards to facilitate interchange of digital data between dissimilar hardware and software systems. Implementation of Navy CALS infrastructure modernization.

3. Phase III (1997 -). Development and application of Integrated Weapon Systems Data Bases (IWSDB) spanning over the entire product life-cycle.

2. Infrastructure Modernization

Infrastructure modernization is focused on three application areas: Engineering Data Automation, Logistic Support Analysis (LSA) Automation and Technical Manual Automation. The major thrust of this modernization effort is to take place during the application of Phase II of the implementation plan.

The objective of this thesis is to address issues related to different DBMS approaches for implementation of IETM systems. This coincides with the modernization effort that strives to automate technical manuals, thereby providing new technology that can support the application of Phase II of the Navy CALS Implementation plan.

V. DON CALS ORIENTED SYSTEMS & DATABASE ENVIRONMENT

A. BACKGROUND

The following sections will briefly describe the existing and planned CALS oriented systems architecture environment in the Navy, while highlighting database architecture aspects of these systems. The word "oriented" is used to emphasize that the description does not necessarily follow the existing Navy organizational definition of CALS systems. Furthermore, since the boundaries between logistics and maintenance systems are not always clearly defined, an exact definition of the term "logistics" (as contained in the CALS acronym) will be avoided at this time, to prevent exclusion of systems that serve both disciplines.

The purpose of this review, therefore, is to identify systems and databases in the logistic/maintenance environment and to identify problems associated with these systems and databases.

B. DON CALS PROCESS ARCHITECTURES

Three process architectures have been identified and described in (DoN, CALS Architecture/Implementation Plan, 1991, pp. 4-24): the Engineering Drawings Process Architecture, the Logistic Support Analysis Record (LASR) Process Architecture and the Technical Manuals Process Architecture. Each process architecture is described in terms of the underlying process, the different manipulators of data (creators, managers and users

of data), the databases and interactions on them and the existing and planned implementations (systems) to support the process architecture. The following details of these process architectures will assist in obtaining a better understanding of the DoN CALS system/data environment and the potential use of these databases:

1. Engineering Drawings

Engineering drawings are created during the design and development processes as the means of documenting product definitions. Along with these drawings, various associated information is provided, such as parts lists, textual descriptive data, indices, configuration control data, etc. (DoN, CALS Architecture/Implementation Plan, 1991, pp. 9-13). The engineering drawings are used not only to support the design and development processes, but for Integrated Logistic Support (ILS), Logistic Support Analysis Records (LSAR), acquisition planning, production cost estimates, product manufacturing, quality assurance, installation and maintenance. Elements of the drawings are used as source data for creating ILS products such as technical manuals, training materials etc. (DoN, CALS Architecture/Implementation Plan, 1991, p. 10).

Until recently, standard media for engineering drawings has been either paper or aperture cards. The Engineering Data Management Information and Control System (EDMICS) is an on-going Navy project designed to automate storage and retrieval of engineering drawings which are maintained as raster images in distributed repositories, thus eliminating the need to store engineering drawings on hard-copy media. An indexing method is used to access the raster image files maintained in optical storage devices.

2. Logistic Support Analysis Record

Logistic Support Analysis (LSA) is an iterative analytical process applied throughout the system acquisition program in order to define supportability related design factors and to ensure development of a fully integrated system support structure (DoD Directive 5000.2, 1991). MIL-STD-1388-1A defines the sub tasks of conducting this analysis (e.g. Program Planning and Control Tasks, Mission and Support System Definition Tasks, etc.).

The Logistic Support Analysis Record (LSAR) defines detailed data elements that identify the logistic support resource requirements of a given system (MIL-STD-1388-2B, 1991, ...). LSAR data is used for conducting LSA tasks, develop ILS products, maintain products and product configuration control and update other management and configuration data bases (MIL-STD-1388-2B, 1991, ...). MIL-STD-1388-2B defines a standard DoD format for a normalized relational database containing all data elements necessary to support the LSA process.

A full implementation of the LSAR relational database, as defined by MIL-STD-1388-2B, has not yet materialized due to limited funding. Currently two proprietary software systems (LEADS, SLIC) are used by different Navy activities to support the LSA process. The SLIC system has limited relational capability to support ad-hoc SQL queries, and is based on a flat-file data architecture. Thus the current state of LSAR data automation can be characterized as suffering from the following problems: lack of software standardization, lack of system connectivity to facilitate data sharing between

sites, and lack of a database architecture to support data integration and flexible data retrieval.

3. Technical Manuals

Technical manuals are developed primarily for the purpose of conducting system maintenance. LSAR data and engineering drawing data are retrieved from the source level LSAR and engineering drawing databases and used by the Technical Manual authoring systems for the purpose of developing technical manuals (DoN, CALS Architecture/Implementation Plan, 1991, pp. 5-8). These authoring systems maintain application level databases to capture technical manual content, format etc., thus providing primarily for print-on-demand capability of technical manuals. However, a far more revolutionary approach to the acquisition and use of technical manuals has been advocated, in the form of Interactive Electronic Technical Manual (IETM). The ITEM concept is the subject of the following chapter.

VI. INTERACTIVE ELECTRONIC TECHNICAL MANUALS

A. BACKGROUND

The increasing complexity of weapon systems developed in recent years has resulted in greater volumes of printed technical manuals to support the maintenance of these systems. However, the overall quality of these manuals has been recognized as being poor (NAVINSGEN, Review of Navy Technical Manual Program, 1984), thereby reducing the quality of the preformed maintenance and decreasing the readiness level of operational systems (aircraft, ships, subsystems, etc.).

One of the major CALS R&D thrusts within the DoD community is to improve maintenance and logistic-support by automating technical manuals. Different systems were developed in the past and other systems are currently being developed for management, printing, distribution and use of technical manuals. Some of these development efforts are unique to specific services while others are Joint Services efforts[2].

---

[2] Examples for management systems are: Air Force Technical Order Management System (AFTOMS), which was later transformed to the Joint Uniform Services Technical Information Service (JUSTIS). Examples for printing systems are: Navy Print On Demand System (NPODS). Examples for delivery/presentation systems are Air Force Integrated Maintenance Information System (IMIS), Navy Technical Information Presentation System (NTIPS).

B. PROBLEMS OF PAPER-BASED TECHNICAL MANUALS

Currently, the main method of delivering technical manual data from the contractors to the DoD is in paper form[3]. To better understand the motivation for development of IETM systems, some of the problems associated with this mode of data delivery are listed below (DTRC-89/007, February 1989, pp. 6-7 and 16-17):

1. Weight and space demands for technical manual libraries.
2. Time required to locate data within a given manual.
3. Requirements to reference other technical manuals, resulting in a mass of paper required to preform maintenance tasks.
4. Low quality updates due to reliance on manual insertion of new correction pages or pen-and-ink corrections made to existing pages.
5. High investment required for training technicians.
6. Rising costs associated with maintaining technical manuals.
7. Low quality maintenance due to the discrepancy between the reading capability of the technician and the language and format of the manuals.
8. Lack of guidance from experienced personnel, thereby resulting in a higher rate of maintenance errors and increased down-time.

The previously outlined problems associated with delivery of technical manuals in paper form, from the perspectives of both the acquiring organization and the end-user,

[3] Alternate automated methods for data delivery have been defined in MIL-STD-1840A, *Automated Interchange of Technical Information.* Although requirements for digital delivery of data have started to appear in recent acquisition contracts, the bulk of data delivered under existing contracts is still in paper-form.

establish the justification for Technical Manual automation. Methods to achieve this objective are detailed in the following section.

C. METHODS FOR TECHNICAL MANUAL AUTOMATION

Three methods for Technical Manual automation have been defined in (DTRC-89/007, February 1989, pp. 13-15). The methods differ by the extent of computer technology application and technical manual data organization:

1. Storage of digitized Technical Manuals which are created in page-oriented form, and are thereafter raster-scanned and stored in digital storage devices. For the purpose of maintaining such a repository, a basic file system with indexing capability will suffice, although usage of a DBMS can provide additional capabilities for conducting queries and searches.

2. Same as the previous method, only that intelligent scanning software is used to identify text and vector graphics that compose the scanned image. Thus the automated system maintains some knowledge of the data content of the technical manual page, enabling advanced techniques of data access. In this case utilization of a DBMS is an essential requirement.

3. Production of frame-oriented technical manuals. In this method text data and image data are created as separate entities and are subsequently maintained as separate entities in the DBMS. Text and images are merged as a result of interactive input by the user, who creates a demand for representation of a new frame on the output device. The data in the frame serves as guidance for the user in performing his current task.

The first two methods are practical for automation of existing technical manuals, whereas the latter would be more cost-effective if used for the development of Technical Manuals for new weapon systems. The Interactive Electronic Technical Manual (IETM)

is based on the previously described concept of a frame-oriented technical manual. More details on the IETM concept can be found in (DTRC-89/007, February 1989, pp. 18-82).

D. SPECIFICATIONS FOR IETM

The Defense Quality and Standardization Office established a Tri-Service ITEM Working Group in 1989 to foster the exchange of ideas and to develop a set of DoD specifications for IETM acquisition. A series of five specifications and handbooks for IETM Acquisition has been drafted by the ITEM Working Group and the David Taylor Research Center, which is the Navy's representative to and the chair of the ITEM Working Group[4].

---

[4] The draft specifications and handbooks consist of:

1. David Taylor Research Center, DTRC Report 90/025, *Proposed Draft Military Handbook Presenting Requirements for an Electronic Display System for Interactive Electronic Technical Manuals (IETMs)*, by Jorgensen E. L., Rainey S. C. and Fuller J. J., July 1990.
2. David Taylor Research Center, DTRC Report 90/026, *Proposed Draft Military Handbook for Preparation of View Packages in Support of Interactive Electronic Technical Manuals (IETMs)*, by Rainey S. C., Jorgensen E. L. and Fuller J. J., July 1990.
3. Tri-Service Working Group for Interactive Electronic Technical Manuals, Draft MIL-M-GCSFUI, *Manuals, Interactive Electronic Technical: General Content, Style, Format, and User-Interaction Requirements*, by Fuller J. J. and others, April 1991.
4. Tri-Service Working Group for Interactive Electronic Technical Manuals, Draft MIL-D-IETMDB, *Data Base, Revisable: Interactive Electronic Technical Manuals, for the Support of*, by Fuller J. J. et al., April 1991.
5. Tri-Service Working Group for Interactive Electronic Technical Manuals, Draft MIL-Q-IETMQA, *Quality Assurance Program: Interactive Electronic Technical Manuals and Associated Technical Information; Requirements for*, by Fuller J. J. et al., April 1991.

One of the specifications contains detailed requirements for an Interactive Electronic Technical Manual Data Base (IETMDB) (MIL-D-IETMDB, April 1991). IETMDB requirements, as well as issues related to the database content and structure, are detailed in the following section.

E. THE INTERACTIVE ELECTRONIC TECHNICAL MANUAL DATA BASE

As with any other database, the IETMDB can be characterized by the data content (i.e. the types of data elements, attributes and relationships between elements, which are maintained in the database) and the database structure type (i.e. the methodology used to implement the DBMS: relational, object-oriented, etc.). The IETMDB content and the IETMDB structure, as well as additional IETMDB functional requirements are detailed below:

1. IETMDB Content

The IETMDB is a complete collection of data elements, attributes and relationships pertaining to a specific weapon system (or other equipment acquired by the Government).

The IETMDB elements are structured in accordance with the hierarchical relationships defined in the Content Data Model[5] (CDM) Data Type Definitions (DTD)[6] and named in accordance with the CDM Data Element Dictionary (Tag Set Descriptions).

2. IETMDB Structure

No structural requirements on the actual Data Base Management System (DBMS) methodology are imposed by MIL-D-IETMDB, i.e. the database can be either a relational or an object-oriented database (MIL-D-IETMDB, April 1991, p. 3).

3. IETMDB Functional Requirements

The following is a list of IETMDB functional requirements. Additional details can be found in (MIL-D-IETMDB, April 1991, pp. 6-12).

1. The ITEMDB can serve as the basis for construction and update of weapon-system electronically displayed ITEMs and automated construction of IETM View Packages[7].

---

[5] The Content Data Model is a specification for a weapon system technical information database. The model was developed by Air Force Human Resources Laboratory (AFHRL-TP-90-10, May 1990), and makes use of the Standard Generalized Markup Language (SGML) specified in (ISO-8879, 1986).

[6] The Document Type Definition (DTD) defines the grammar of the tag language used within a document (i.e. the names of tags which label the data items and the structure of the data. A good example for use of SGML and DTD can be found in (AFHRL-TP-90-10, May 1990, pp. 8-14).

[7] An IETM View Package is defined as "a fully organized and formatted item of computer-processible Technical Information derived from an IETMDB and capable of interactive electronic display to an end user by means of an Electronic Display System (EDS)" (DTRC Report 90/026, July 1990, p. 13).

2. The IETMDB can provide direct access to logistic-support information related to a specific weapon system.

3. The IETMDB, or portions of it, can be interchanged by means of standardized formats and procedures.

4. The IETMDB shall not contain format directions for arrangement of text and graphics on a display screen (i.e. the data will be "format-free").[8]

As mentioned above, the IETMDB specification does not contain a requirement for usage of a specific DBMS methodology. One of the goals of this thesis is to investigate the application of OODBMS methodology in the implementation of IETMDB. This topic is the subject of the following chapter.

[8] In page-oriented technical manuals a strong binding exists between the page data (text and images) and the page format. This binding is eliminated in the frame-oriented technical manual concept.

VII. DESIGN OF AN OBJECT ORIENTED IETMDB

A. INTRODUCTION

The purpose of this chapter is to demonstrate how an IETMDB can be designed by utilizing object-oriented design concepts and methodology. The first section provides a detailed description of the Content Data Model (CDM), which is used by the MIL-D-IETMDB specification to describe the technical information elements and their relationships. The second section maps the requirements of the CDM to the various characteristics of the Object-Oriented Paradigm (OOP). The third section contains an actual design of an Object-Oriented class hierarchy for the implementation of IETMDB.

B. CDM CHARACTERISTICS AND STRUCTURE

The following is an outline and brief description of the CDM characteristics and structure. Additional details can be found in (MIL-D-IETMDB, April 1991)(Caporlette, 1991).

1. General Characteristics

The CDM was developed to represent technical information elements and their relationships in an integrated database environment, thus providing capabilities that were non-existent in earlier forms of non-integrated flat-file technical data representations. The CDM main capabilities are (Caporlette, 1991):

1. Elimination of data redundancy. This is achieved by tagging technical data according to content, and not according to format or structure. Figure 1 provides a comparative view of these approaches whereas Figure 2 depicts the non-redundant referencing capability provided by the CDM[9].

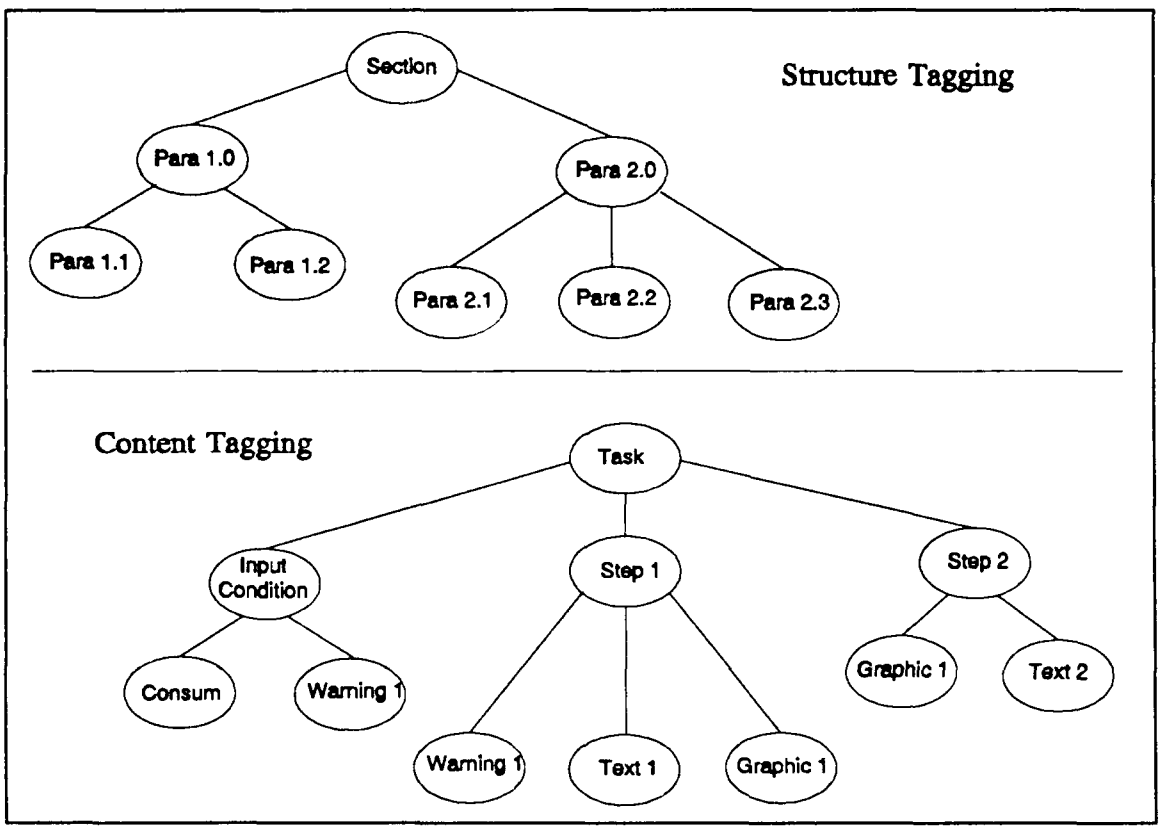

Figure 1 The Structure and Content Tagging Approaches

2. Creation of links between data elements that have unique relationships. This capability supports establishing relational links to access specific data, as depicted in Figure 3. Thus instead of the common reference to data that is used in hard-copy technical manuals (e.g. "see section 1.2 ...") the user is provided with the capability to actually access the data when using the appropriate device, namely the IETM.

[9] Figures 1 through 4 were adapted from (Caporlette, 1991).

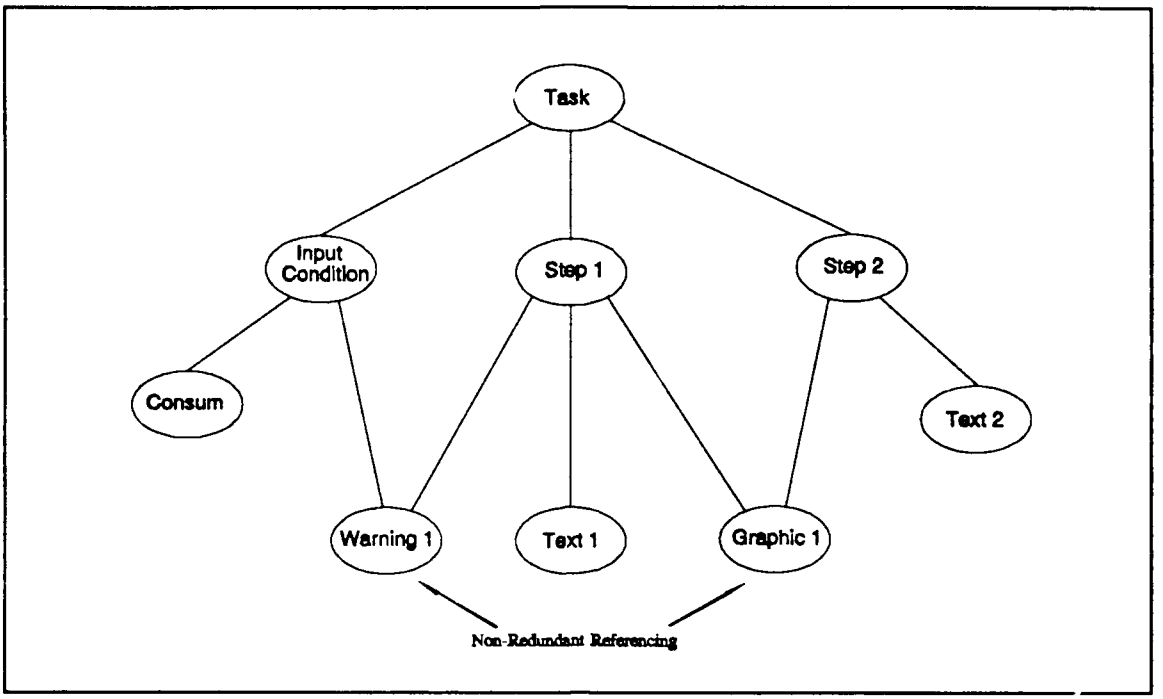

Figure 2 Non-Redundant Referencing

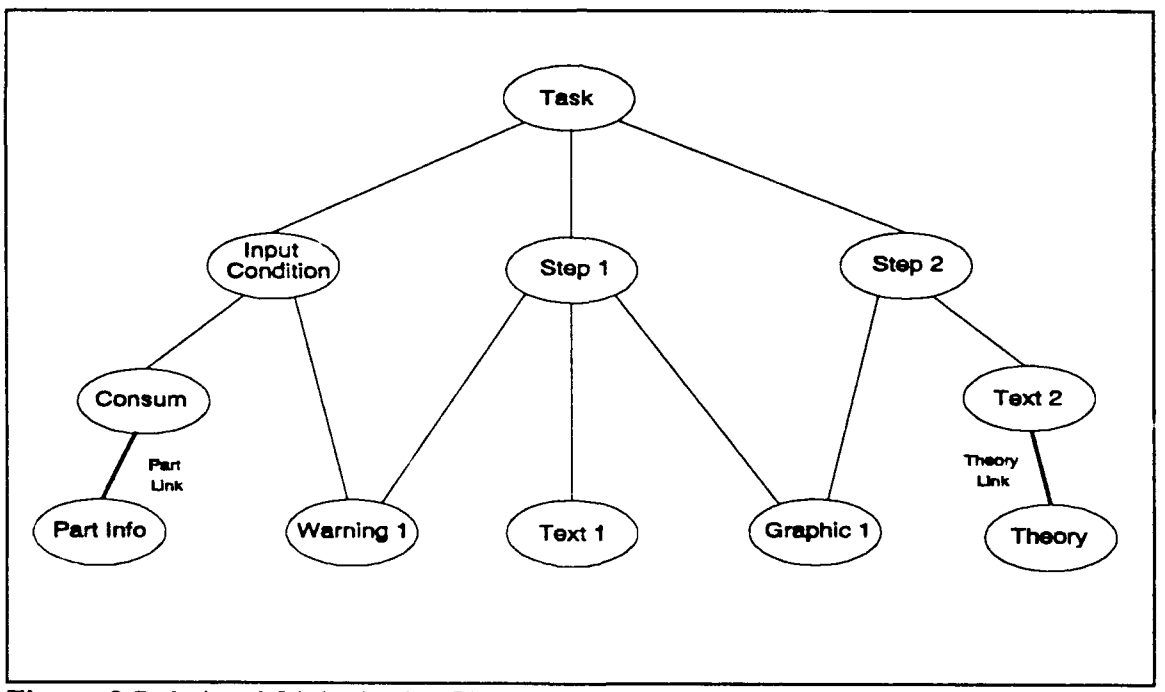

Figure 3 Relational Links in the CDM

3. Provide capability for dynamic selection of technical information based on variables dependant upon usage scenario. An example, in which the variable is the expertise level of the technician, is depicted in Figure 4.

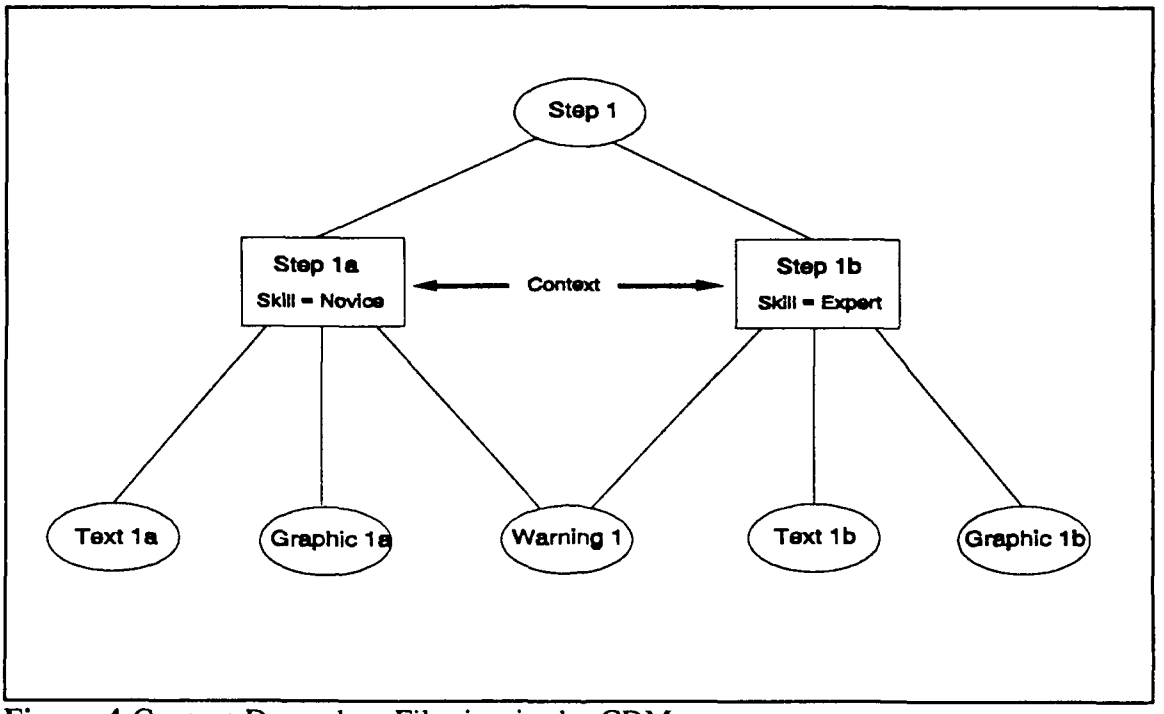

Figure 4 Context Dependant Filtering in the CDM

4. Provide capability for user interaction and branching, as depicted in Figure 5. This is a fundamental requirement of the IETM concept, because of the unpredictability of the maintenance task flow.

As mentioned earlier, these capabilities were not provided by earlier models of technical data. The dual layered CDM structure, which provides these capabilities, is detailed in the following subsections.

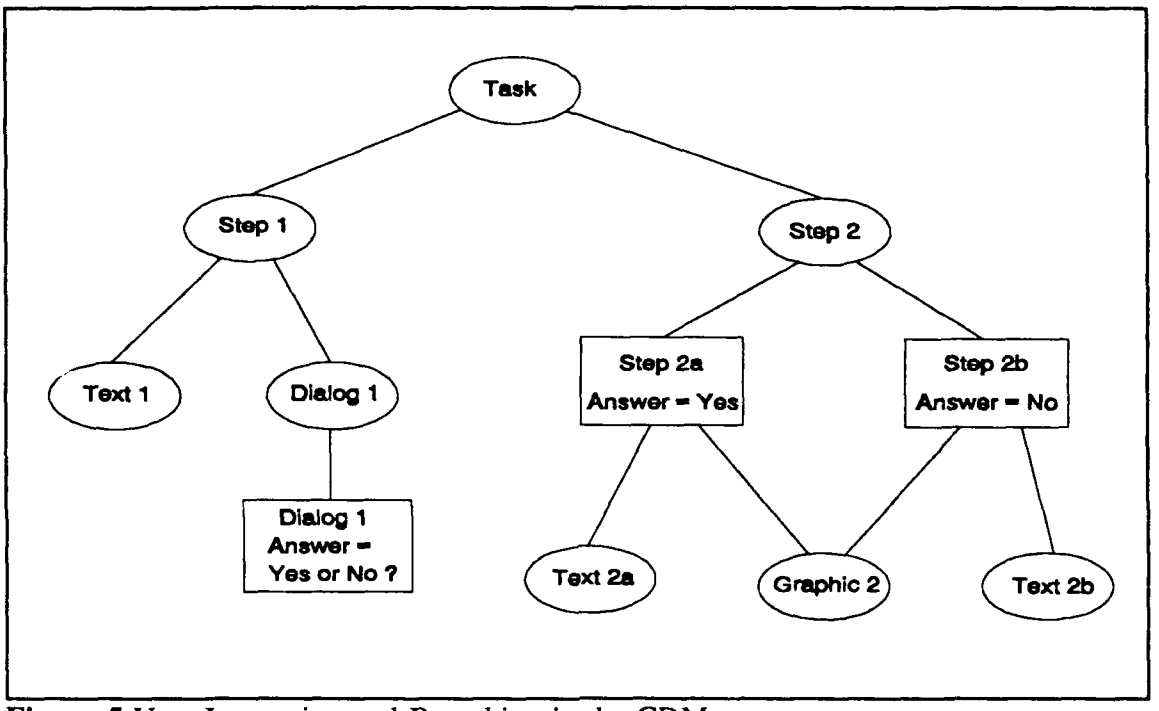

Figure 5 User Interaction and Branching in the CDM

2. General Structure

The basic structure of the CDM consists of two layers: the Generic Layer, which defines the general characteristics of the CDM and the structures which are common to all applications, and the Content Specific Layer which defines content specific structures and their relationships for a given application.

The Generic Layer consists of templates, linking elements, primitive elements and context filtering elements. The Content Specific Layer uses the Generic Layer templates and elements to define the application elements and relationships.

The overall CDM structure is depicted in Figure 6.

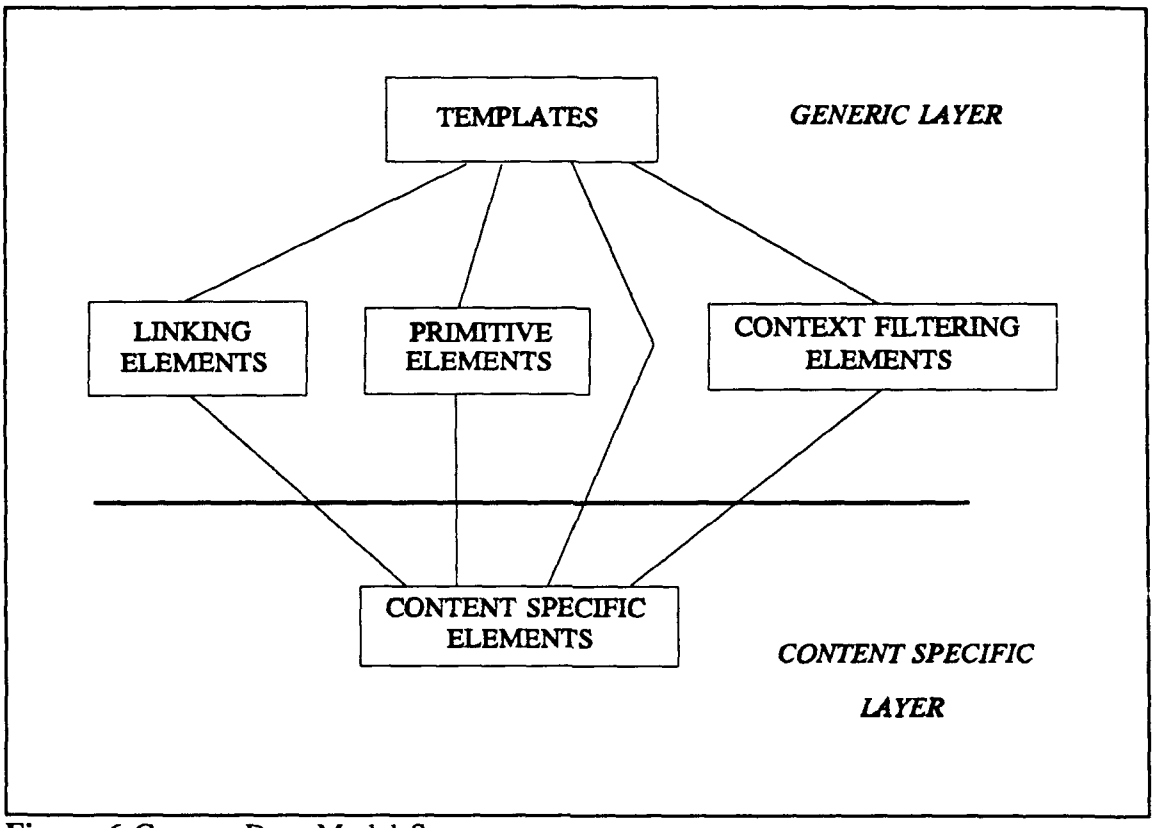

Figure 6 Content Data Model Structure

3. Elements

Elements are used to capture data. A SGML Data Type Definition (DTD) declaration of an element consists of three parts: the element name, requirements for beginning and end tags[10] and the definition of the element structure (sometimes referred to as the element content model). The element structure definition details either the elements that make up the specific element (i.e. identify the sub-components) or the actual data of the element. Elements can have multiple occurrences and can have attributes associated with them.

---

[10] The beginning and end tags are part of the SGML syntax.

For the purpose of providing examples of CDM structures throughout this chapter, an abbreviated description of SGML syntax and examples of usage of element declarations, entity declarations and attribute list declarations[11], is contained in Table 1 through Table 3. Additional details on the SGML syntax can be found in MIL-M-28001A, July 1990 (Appendix A). Semantics of the definitions of CDM structures can be found in MIL-D-IETMDB, April 1991 (Appendix A).

[11] Elements and attributes describe the actual data. Entities, on the other hand, are merely a SGML language mechanism for string substitution. See details in Table 1.

TABLE 1 SGML SYNTAX FOR ELEMENT DECLARATION

Element Declaration

Syntax:

<!ELEMENT element_name [tag_minimization] element_structure >

Examples:

<!ELEMENT aircraft - - (body , system+)>
<!ELEMENT body - o (fuselage & wings & tail_section)>
<!ELEMENT system - o (hydraulic+ & navigation* & armament?)>

SGML syntax	Explanation
<!ELEMENT ... >	Element declaration
- o	Requirements for start and end tags minimization: first position = start tag, second position = end tag - = required, o = optional
(...)	Grouping of subcomponents
* + ?	Occurrence indicators: * = zero or more, + = exactly one, ? = zero or one
, | &	Connectors: , = order of elements (sequence) | = only on element of group is used (or) & = elements may occur in any order (and)

Examples explanations:

The aircraft element consists of 2 subcomponents: a (single) body subcomponent and one or more systems subcomponents. The system element has at least one hydraulic subcomponent, zero or more navigation subcomponents and possibly a single armament subcomponent.
The usage of the aircraft element requires that the aircraft data be preceded by a start tag and succeeded by an end tag, whereas the usage of the body element requires only that a start tag precede the body data.

TABLE 2 SGML SYNTAX FOR ENTITY DECLARATION

Entity Declaration	
Syntax: <!ENTITY entity_name entity_text> **Examples:** <!ENTITY dod "Department of Defense"> <!ENTITY % text "(PCDATA)"> **Comments:** The entity declaration serves as a string substitution mechanism	
SGML syntax	**Explanation**
<!ENTITY name "text">	Declaration of general entity. Declared in DTD and referenced in the document instance
<!ENTITY % name "text">	Declaration of parameter entity. Declared in DTD and referenced in DTD (mechanism for shorthand). Usage reference is %name.

4. Templates

Templates define a set of semantic rules for creating either generic elements or content specific elements. The definition of the element is not restricted to the content of the template used to define it, i.e. additional semantic rules can be added to those defined by the template. The templates provide structures that can be used for definition of composite elements, execution of context dependant filtering, the capture of user interaction sequences and execution of conditional branching and iteration.

TABLE 3 SGML SYNTAX FOR ATTRIBUTE LIST DECLARATION

Attribute List Declaration

Syntax:

<!ATTLIST element_name attribute_definition_list>

Examples:

```
<!ATTLIST aircraft
        type ( fighter | helicopter | transport | undefined ) "undefined"
        branch ( af | navy | army | mc | cg ) #REQUIRED
        tail_no NUMBER #REQUIRED >
```

SGML syntax	Explanation
<!ATTLIST ... >	Attribute list declaration
"value"	Definition of a default value
#KEYWORD (e.g. #REQUIRED)	Keywords for attribute specification requirements: #REQUIRED = a specified value is required #IMPLIED = the value is implied by the application #CURRENT = required on first usage of that attribute for that element, otherwise defaults to previous value #CONREF = filled in only when the element's content model is empty
KEYWORD (e.g. NUMBER)	Keywords for attribute declared values: CDATA = character data NUMBER = all digits NAME = beginning with alphabetic character, then either alphanumeric, '-' or '.' ID = a unique identifier IDREF = a reference to an ID NUTOKEN = begins with digit and contains name characters ENTITY = a reference to an externally declared entity

Five templates are defined in the CDM Generic Layer: The Node template, the Node Alternatives template, the Node Sequence template, the If Node template and the For Node template. Following are the details of each template:

a. Node

The Node (NODE) contains the content of the technical information, context filtering preconditions and postconditions and link elements that provide for cross-referencing to other technical information (nodes). The link element provides for definition of composite structures of nodes, thus creating an implied hierarchy of technical information elements.

Table 4 contains the definition for the NODE template and an example[12] for its use. The example describes the content specific "system" element, which employs the NODE template from the generic layer. The "system" element is used to describe any component which has technical information associated with it. The system-subsystem hierarchy of an aircraft (i.e. vehicle/system/subsystem/subassembly) can be modeled in the CDM by using the content specific "system" element to define any component in the vehicle hierarchy. Figure 7 shows the relationship between the aircraft technical information hierarchy, and its representation in the CDM context specific layer.

[12] Content specific examples given in this chapter are taken from (MIL-D-IETMDB, April 1991, Appendix B) which contains the content specific element DTDs for display of technical information for an O-level maintenance technician.

TABLE 4 THE NODE TEMPLATE

Template name: **NODE**

Generic template:

```
<!ELEMENT "NODE" - o ( precond*, link*, ( NODE | NODE_ALTS |
                NODE_SEQ )*, postcond* )>

<!ENTITY % a.node
            "id      ID        #IMPLIED
             name    CDATA     #IMPLIED
             type    CDATA     #IMPLIED
             itemid  CDATA     #IMPLIED
             cdm     NAME      #FIXED     'node'
             ref     IDREF     #CONREF" >
```

Semantic interpretation:

1. The content specific element utilizing the NODE template may contain preconditions, which will be evaluated at presentation time. The NODE will be presented if all conditions evaluate to true.
2. The element may contain relational links to other elements.
3. The element may contain subcomponents that employ the NODE, NODE_ALTS, or NODE_SEQ templates.
4. The element may contain postconditions which record presentation events.

Usage example:

```
<!ELEMENT system - o ( precond*, link*, %system;*, %descinfo;*, %task;*, %partinfo;*,
                       %faultinfo;* )>
<!ATTLIST system
        %a.node; >
```

Example comments:

1. The context specific "system" element employs the NODE template from the generic layer. The "system" element contains a list of preconditions which define the elements applicability, relational links to other elements, sub-system elements (by means of the %system;* entity) and descriptive, task, part, and fault information about the system.
2. Although the NODE template allows for postconditions as well, these are optional and are not used in the context of the example.

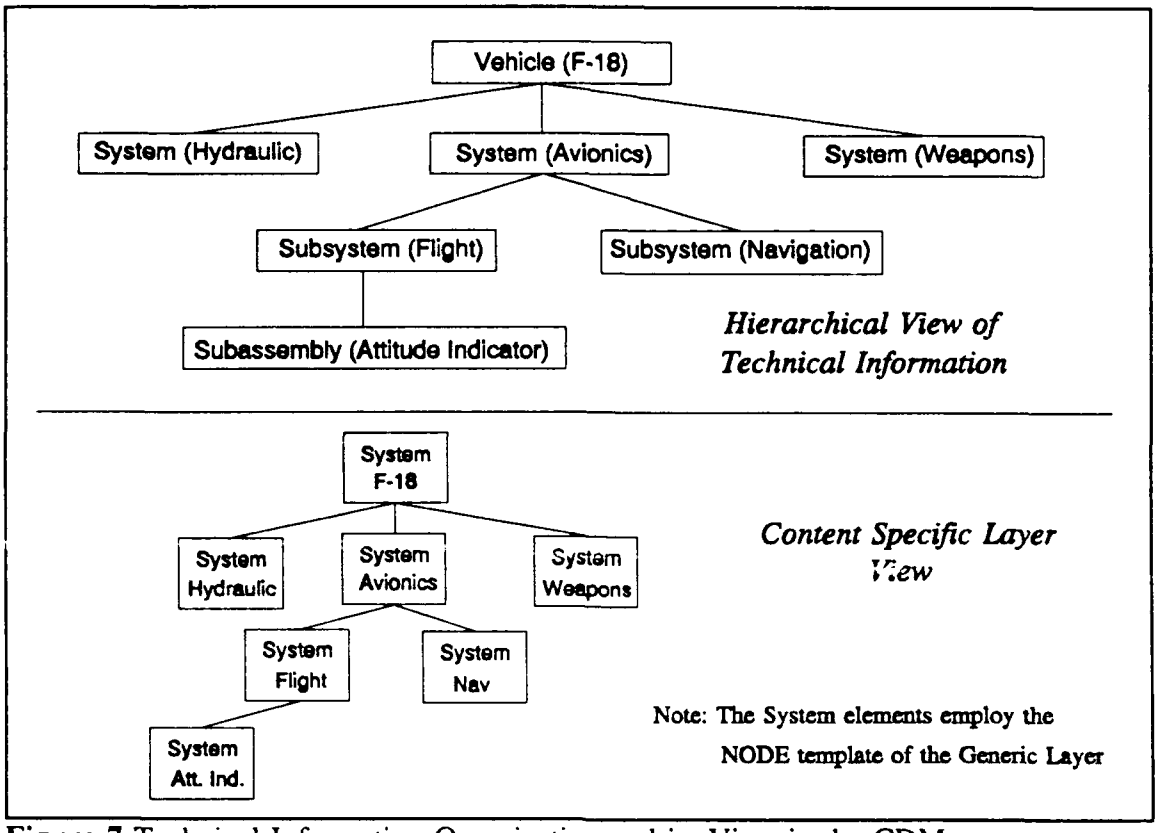

Figure 7 Technical Information Organization and its View in the CDM

b. Node Alternatives

Node Alternatives (NODE_ALTS) is a list of mutually exclusive nodes, grouped together by the fact that they apply to different contextual situations. The content specific layer NODE_ALTS element is a reference to a set of nodes that might apply in different situations. No hierarchy is implied between the generic NODE_ALTS element and the content specific NODE element.

Table 5 contains the definition for the NODE_ALTS template and an example for its use. The example describes the content specific "descinfo_alt" element, which employs the NODE_ALT template from the generic layer. Figure 8 depicts the

relationship between the actual descriptive information contained in the technical manual, and the way it is modeled in the content specific layer of the CDM using the NODE_ALT template from the generic layer.

TABLE 5 THE NODE ALTERNATIVES TEMPLATE

Template name: **NODE_ALTS**
Generic template: <!ELEMENT "NODE" - o (NODE)+ > <!ENTITY % a.node_alts "id ID #IMPLIED cdm NAME #FIXED 'node_alts' ref IDREF #CONREF" >
Semantic interpretation: 1. The content specific element utilizing the NODE template must contain components that employ the NODE template. 2. The components must be of the same element type and at the same level in the hierarchy. 3. The each alternative, the NODE whose precondition evaluates to "true" will be presented. 4. The preconditions must be mutually exclusive such that no more than one NODE will have a precondition that evaluates to "true". 5. There need not be an applicable component for every possible situation.
Usage example: <!ELEMENT descinfo_alts - o (descinfo)+ > <!ATTLIST descinfo_alts %a.node_alts; >
Example comments: 1. The "descinfo_alts" element is uses the NODE_ALTS template to facilitate context filtering of descriptive information.

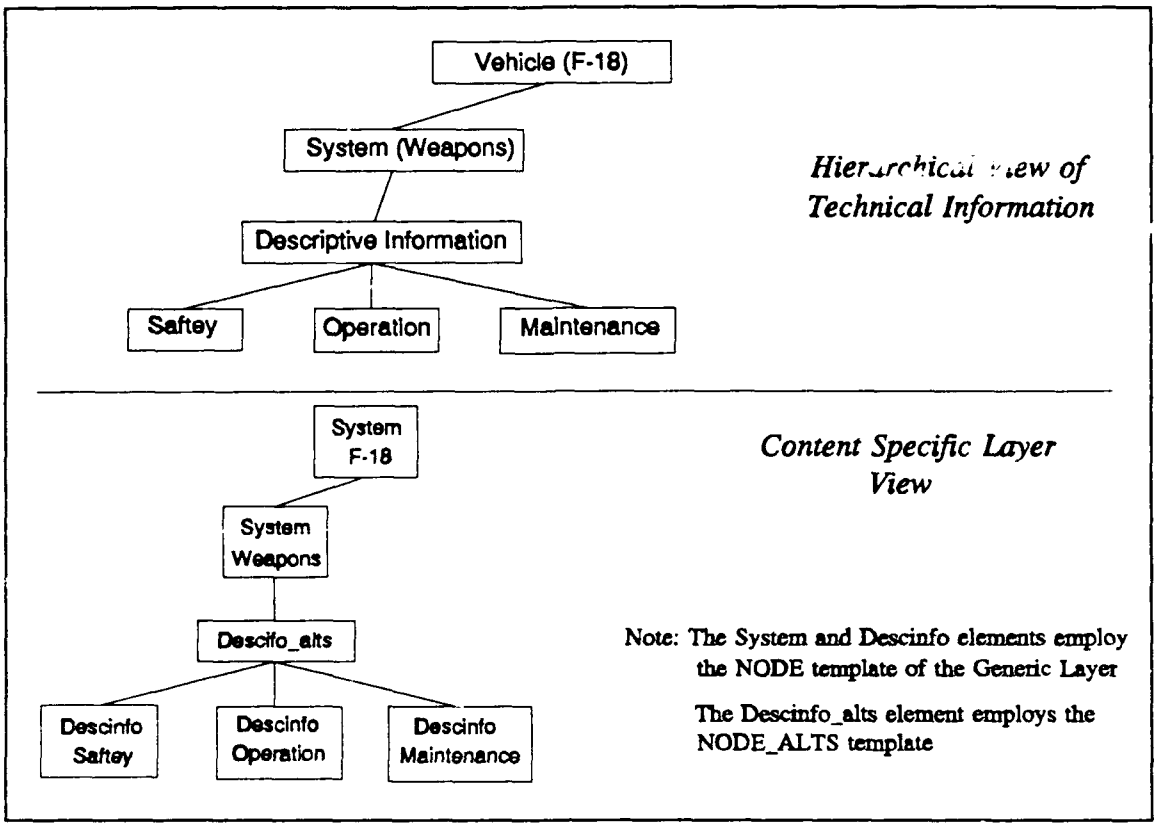

Figure 8 Using the NODE_ALT Template

c. Node Sequence

The Node Sequence (NODE_SEQ) template is used for definition of content specific elements that capture user interaction sequences. The components of a NODE_SEQ are elements that use the NODE, NODE_ALTS, IF_NODE or FOR_NODE templates. The components of a NODE_SEQ are traversed in their order of appearance, to include branching and iteration as implied by the different templates.

Table 6 contains the definition for the NODE_SEQ template and an example of its use. The example describes the content specific "step_seq" element, which employs the NODE_SEQ template from the generic layer. Figure 9 depicts the

relationship between the actual descriptive information contained in the technical manual, and the way it is modeled in the content specific layer of the CDM using the NODE_SEQ template from the generic layer.

TABLE 6 THE NODE SEQUENCE TEMPLATE

Template name: **NODE_SEQ**
Generic template: <!ELEMENT NODE_SEQ - - (NODE | NODE_ALTS | IF_NODE | FOR_NODE)+ > <!ENTITY % a.node_seq "id ID #IMPLIED cdm NAME #FIXED 'node_seq' ref IDREF #CONREF" >
Semantic interpretation: 1. The content specific element utilizing the NODE template must contain components that employ the NODE, NODE_ALTS, IF_NODE or FOR_NODE templates. 2. The components of a NODE_SEQ are traversed in the order of appearance.
Usage example: <!ELEMENT step_seq - o (step | step_alts | if_step | for_step | task | task_alts)+ > <!ATTLIST step_seq %a.node_seq; >
Example comments: 1. The "step_seq" element uses the NODE_SEQ template to provide capability to create sequences of steps and tasks that comprise a maintenance procedure.

d. If Node

The If Node (IF_NODE) template is similar to an if-then-else statement in programming languages. It contains a precondition which is evaluated, and according

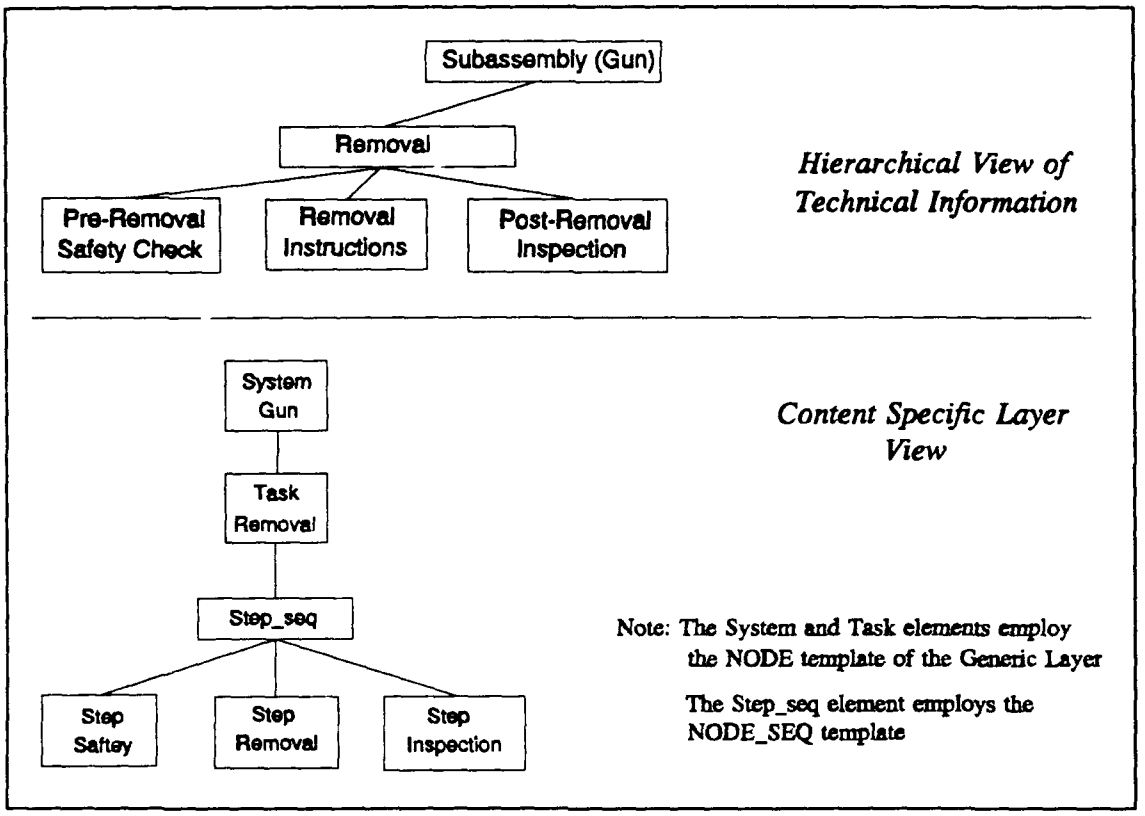

Figure 9 Using the NODE_SEQ Template

to the outcome a NODE_SEQ (which is a then/else component of the IF_NODE) might be traversed.

Table 7 contains the definition for the IF_NODE template and an example for its use. The example describes the content specific "if_step" element, which employs the IF_NODE template from the generic layer. Figure 10 depicts the relationship between the actual descriptive information contained in the technical manual, and the way it is modeled in the content specific layer of the CDM using the IF_NODE template from the generic layer.

TABLE 7 THE IF NODE TEMPLATE

Template name: **IF_NODE**
Generic template: <!ELEMENT IF_NODE - - (precond, NODE_SEQ, NODE_SEQ?) > <!ENTITY % a.if_node "id ID #IMPLIED cdm NAME #FIXED 'if_node' ref IDREF #CONREF" >
Semantic interpretation: 1. The precondition will be evaluated during presentation. 2. If it evaluates to "true", the first NODE_SEQ will be traversed. 3. Otherwise, if there exists a second NODE_SEQ, it will be traversed. 4. If a second NODE_SEQ doesn't exist, no action will be taken.
Usage example: <!ELEMENT if_step - o (precond, step_seq, step_seq?)> <!ATTLIST if_step %a.if_node; >
Example comments: 1. The "if_step" element uses the IF_NODE template to provide capability for conditional selection of steps, to be executed in a specific maintenance procedure.

e. For Node

The For Node (FOR_NODE) provides the capability of iterating over a NODE_SEQ, in a similar manner to the "for loop" of a programming language. It's components are preconditions and postconditions to initialize, test and update the loop control variable, as well as a NODE_SEQ which constitutes the "loop body".

Table 8 contains the definition for the FOR_NODE template and an example of its use. The example describes the content specific "for_step" element, which

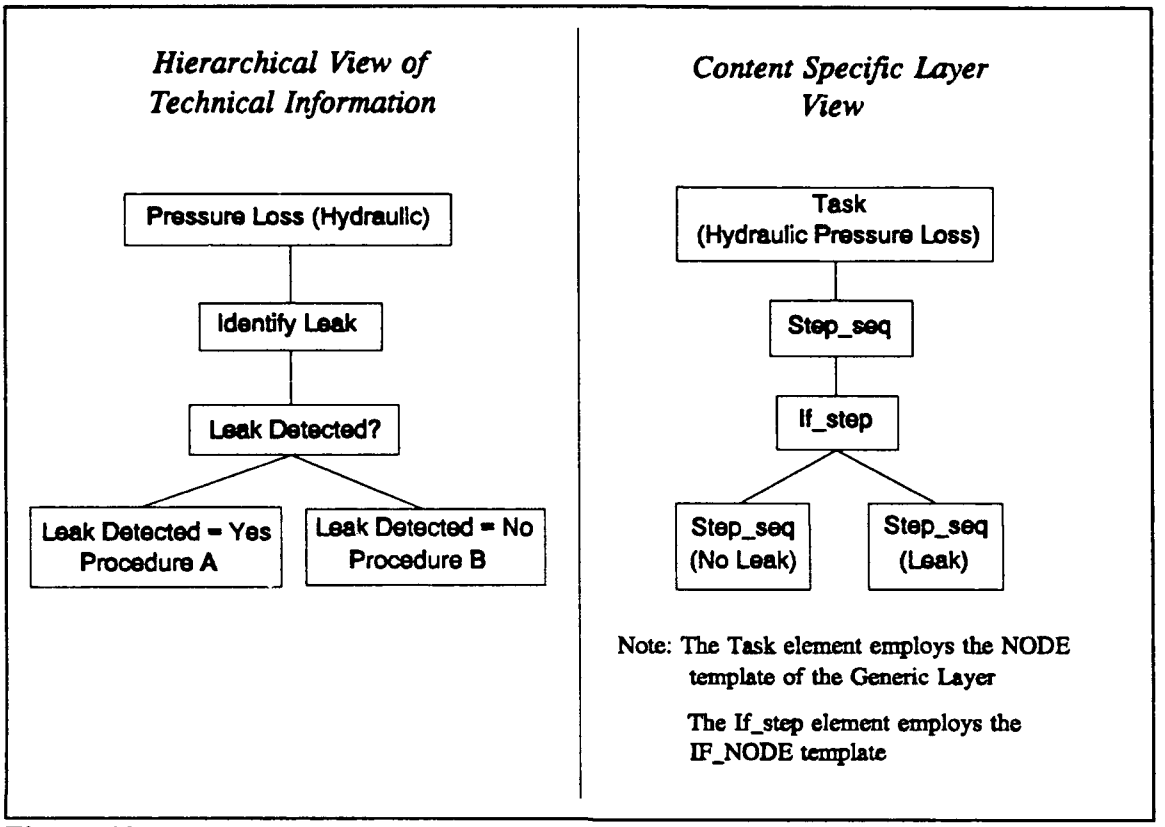

Figure 10 Using the IF_NODE Template

employs the FOR_NODE template from the generic layer. Figure 11 depicts the relationship between the actual descriptive information contained in the technical manual, and the way it is modeled in the content specific layer of the CDM using the FOR_NODE template from the generic layer.

5. Linking Elements

The generic layer link element provides the capability to create links between different elements of the CDM. The linking mechanism employed by the CDM is based on the HyTime standard (ISO/IEC 10744, April 1991). The link element attributes contain identifiers that point to either a CDM element or a location element that resolves

TABLE 8 THE IF NODE TEMPLATE

Template name: **IF_NODE**
Generic template: <!ELEMENT IF_NODE - - (precond, NODE_SEQ, NODE_SEQ?) > <!ENTITY % a.if_node "id ID #IMPLIED cdm NAME #FIXED 'if_node' ref IDREF #CONREF" >
Semantic interpretation: 1. The precondition will be evaluated during presentation. 2. If it evaluates to "true", the first NODE_SEQ will be traversed. 3. Otherwise, if there exists a second NODE_SEQ, it will be traversed. 4. If a second NODE_SEQ doesn't exist, no action will be taken.
Usage example: <!ELEMENT if_step - o (precond, step_seq, step_seq?)> <!ATTLIST if_step %a.if_node; >
Example comments: 1. The "if_step" element uses the IF_NODE template to provide capability for conditional selection of steps, to be executed in a specific maintenance procedure.

to the desired data. Seven different location elements are defined in the CDM: the external element pointer, the element location pointer, the data entity location pointer, the data location pointer, the aggregate location pointer, the generated location pointer and the span location pointer. Additional details on the HyTime standard can be found in ISO/IEC 10744, April 1991.

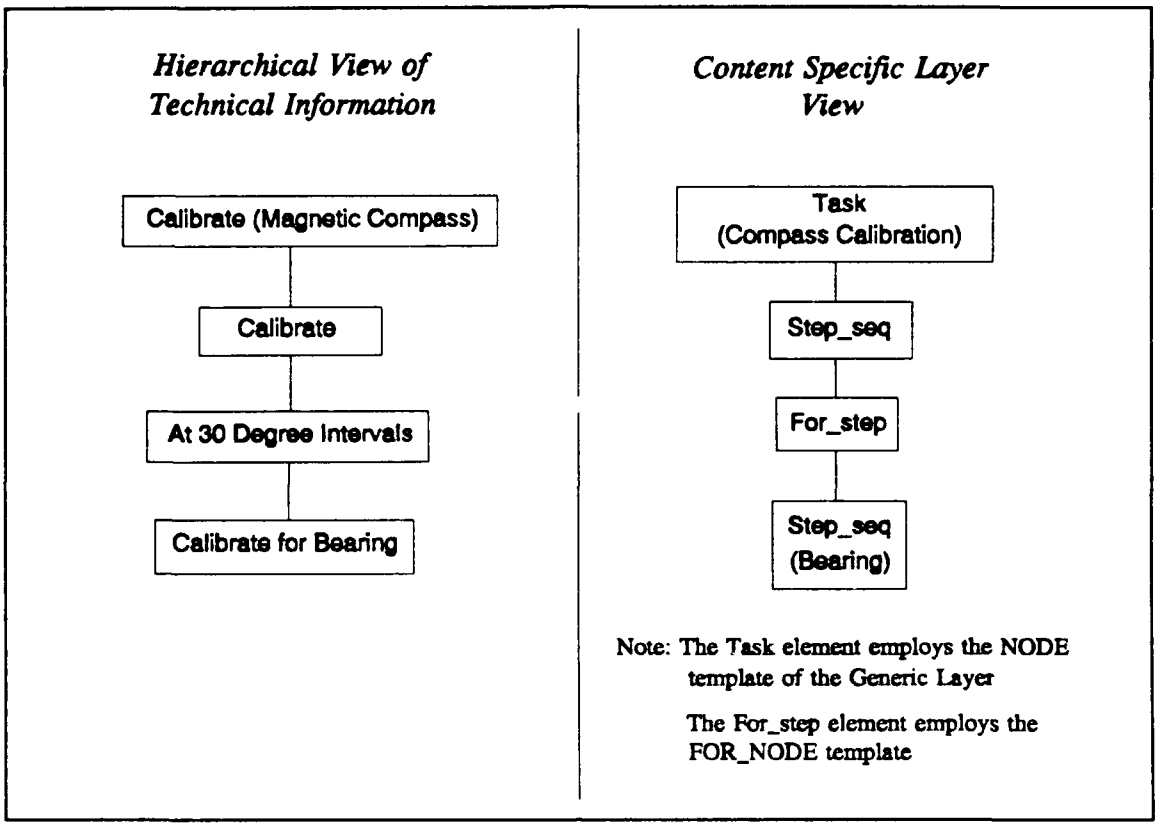

Figure 11 Using the FOR_NODE Template

6. Primitive Elements

The primitive elements in the CDM generic layer consist of the basic text element (which is a text string of parsable character data), table elements (table, column header, entry), graphics elements (graphic, graphic primitive), audio, video and process elements (audio, video, process, parameter) and dialog elements (dialog, fill-in, menu, prompt, choice, selection).

7. Context Filtering Elements

Context filtering elements provide the capability to present to the user only the information that applies to his session (preconditions) or record presentation events

TABLE 9 THE FOR NODE TEMPLATE

Template name: **FOR_NODE**

Generic template:

```
<!ELEMENT FOR_NODE - - ( postcond, precond. postcond, NODE_SEQ ) >

<!ENTITY % a.for_node
              "id     ID        #IMPLIED
               cdm    NAME      #FIXED     'for_node'
               ref    IDREF     #CONREF" >
```

Semantic interpretation:

1. At the beginning of the loop the first postcondition is evaluated and the value is assigned to the specified property.
2. The precondition is evaluated, and if it evaluates to "true" the NODE_SEQ is traversed.
3. At the end of each iteration the second postcondition is evaluated and the value is assigned to the specified property.
4. If the precondition evaluates to anything but "true", the loop is terminated.

Usage example:

```
<!ELEMENT for_step - o ( postcond, precond, postcond, step_seq )>
<!ATTLIST for_step
        %a.for_node; >
```

Example comments:

1. The "for_step" element uses the FOR_NODE template to provide capability for iterating over a series of steps.

for later filtering (postconditions). This capability is useful for training, recording maintenance activities and other control purposes.

8. Content Specific Elements

The content specific elements define those elements that are specific to the application.

Appendix B of MIL-D-IETMDB contains the content specific elements DTD for display of technical information for an O-level maintenance technician. Examples for content specific elements are: descriptive information element, paragraph element, task element, person element, equipment element, fault information element, etc. Other examples can be found in Table 4 through Table 8 and in (MIL-D-IETMDB, April 1991, Appendix B).

C. OBJECT-ORIENTED MAPPING OF THE CDM

The MIL-D-IETMDB specification requires that the database elements be structured according to the hierarchical relationships defined in the CDM DTDs and named in accordance with the CDM Data Element Dictionary (Tag Set Descriptions) (MIL-D-IETMDB, April 1991, p. 6). In view of these requirements the following guidelines were constructed for mapping CDM structures to Object-Oriented Paradigm (OOP) structures:

1. Mapping CDM Structures to Object Classes

As mentioned earlier the CDM consists of two layers: the generic layer which defines general characteristics and generic structures which are common across all CDM applications, and the content specific layer which defines the content specific structures for a given application. Implicit in this structure are requirements for code reusability and code sharing, which the OOP provides for by means of abstraction, inheritance and polymorphism (Wu, 1991).

The generic layer structures map into two types of classes: abstract (formal) classes (i.e. classes that have no instances, and actually serve as encapsulators of common

structure and behavior to be inherited by other classes), and "regular" object classes (i.e. classes that have instances). The generic layer templates map to the former, whereas all other generic layer elements (i.e. linking, primitive and context filtering elements) map to the latter. For example, the NODE template:

```
<!ELEMENT "NODE" - o ( precond*, link* ( NODE | NODE_ALTS |
                                NODE_SEQ )*, postcond* )>
<!ENTITY % a.node
          "id       ID       #IMPLIED
           name     CDATA    #IMPLIED
           type     CDATA    #IMPLIED
           itemid   CDATA    #IMPLIED
           cdm      NAME     #FIXED  'node'
           ref      IDREF    #CONREF" >
```

maps to the abstract class Node, whereas the generic layer primitive element "prompt":

```
<!ELEMENT prompt - o ( %text; | %graphic; )>
<!ATTLIST   prompt
           id       ID       #IMPLIED
           ref      IDREF    #CONREF >
```

will map to a "regular" class named Prompt, which has instances. The topic of defining the instance variables for these classes is addressed in the following subsection.

The content specific layer defines the content specific elements used by the application. Each content specific element constitutes a separate class. For example, the content specific equipment element:

```
<!ELEMENT equip - o ( precond*, link*, %equip;* )>
<!ATTLIST equip
      %a.node;
      cage      CDATA  #IMPLIED
      icc       CDATA  #IMPLIED
      nsn       CDATA  #IMPLIED
      qty       CDATA  #IMPLIED >
```

will map to the Equip class which will define the structure and behavior of all equipment objects. The mapping of CDM structures to OOP structure is depicted in Figure 12.

2. Mapping CDM Structures to Instance Variables

Instance variables are the attributes of the object. It follows that the element attributes map to the instance variables. In the previous example the "equip" element had two groups of attributes. The first group composed of those attributes defined in the NODE template and associated with the "equip" element by means of the entity reference "a.node;". These attributes are directly inherited from the Node class. The second group contains four additional attributes that describe the type and number of equipment items required for a certain task. These attributes map to the Equip class instance variables.

Instance variables are objects of other classes and thereby provide the mechanism for defining the object structure (i.e. subcomponents[13]). The element structure, on the other hand, is defined by the element subcomponnents. For example, the "table" element:

[13] This characteristic of OOP is known as aggregation.

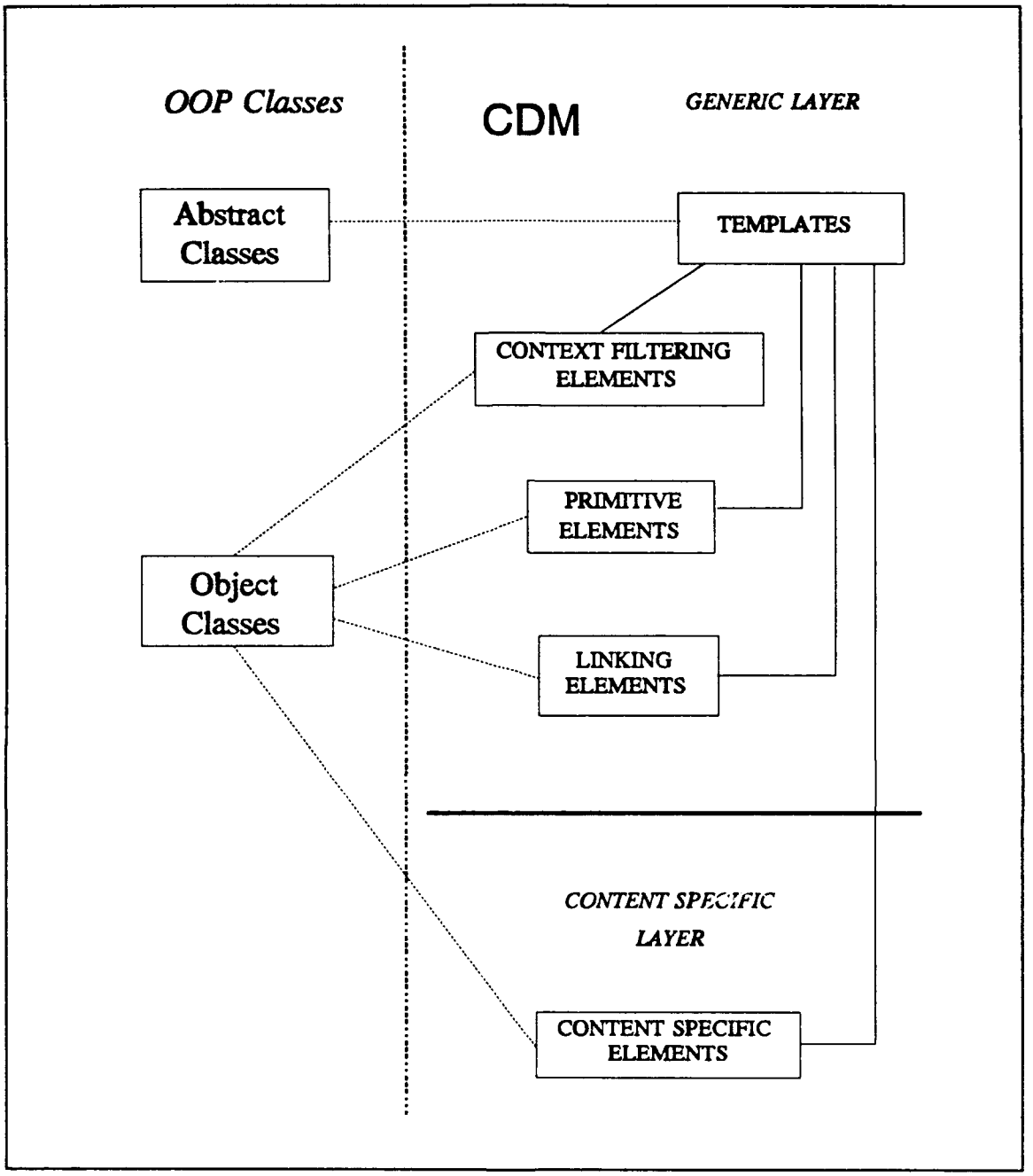

Figure 12 Mapping CDM Structures to OOP Classes

```
<!ELEMENT table - o ( precond*, link*, ( colhddef*, entry+ )+ )>
<!ATTLIST table
        %a.node; >
```

is an aggregation of preconditions, links, and at least one column entry that may have a header. These objects define the structure of the Table class, and are therefore mapped to the Table class instance variables as well. This coincides with the notion that the "table" object need not know about the internal structure of its instance variables (which could be a single entry object, a collection of entry objects, etc.).

3. Mapping Multiple Occurrences of Subelements

As mentioned earlier, the element structure definition allows for multiple occurrences of subelements. For example, the "table" element can have zero or more preconditions and at least one entry[14]. To capture this dat, the Table class object will have a "precond" instance variable, which will be set to nil if no preconditions exist or be set to a given sequence of preconditions otherwise. It will be the responsibility of the loading program to create the correct objects according to the DTD definition of the element structure[15].

4. Mapping Subelement Ordering

The element structure contains connectors that define rules for ordering of subelements[16]. This knowledge of the correct order (as defined in the DTD) is needed by the loading program in order to create instances of the correct class to capture the data. The order of the subelements will not be maintained explicitly by the corresponding

---

[14] See definition of multiple occurrences in Table 1.

[15] See discussion on the software environment in Section D, Subsection 3.

[16] See definition of connectors in Table 1.

database object, but will be implicit in the class type used by the loading program to bind the data to the specific instance variable.

For example, the "|" connector defines an exclusive-or relationship among element subcomponnents of the "choice" element:

```
<!ELEMENT choice - o ( ( %text; | %graphic; ), ( postcond+ | %dialog;) )>
<!ATTLIST choice
        id      ID              #IMPLIED
        ref     IDREF           #CONREF
        default     ( Yes | No )    'No' >
```

It will be the responsibility of the loading program to create either a text object or a graphics object, when a new choice object is created. The choice object will have two instance variables to capture the two subcomponnents of the "choice" element.

D. AN OBJECT CLASS HIERARCHY FOR IETM IMPLEMENTATION

1. General Structure of the IETM Object Class Hierarchy

When designing an Object Class Hierarchy one has to take into consideration two fundamental parameters: the overall functionality provided by each class, on one hand, and the class extensibility characteristics, on the other hand. While striving to maximize both, it is important to note that in certain cases maximizing functional capabilities will not necessarily increase the probability of reuse, since construction of a clear view of the class behavior might become difficult due to an excessive number of methods. What is needed therefore is a balance between functionality and adaptivity.

A class hierarchy design based on a layered structure provides classes that have a good balance between functionality and adaptivity, such that the ease of their reuse is maximized. One such approach is to create a protective layer between the low level system classes that implement hardware specific characteristics and the application classes that implement the user requirements. The protective layer is called the Subject Layer or the Base/Application Layer and its merits were discussed in (Wu, 1991). In accordance with this approach, the following layers have been defined for an object class hierarchy for IETM implementation:

1. Physical Layer - This layer will provide the low level functionality which is mostly system dependant (e.g. file access, key searches, etc.).

2. Subject Layer - This layer will provide the data view defined by the CDM, i.e. all the data elements defined in the CDM (MIL-D-IETMDB, 1991, Appx A-B) will have respective objects in this layer.

3. Application Layer - This layer will consist of object classes to provide the specific functionality defined by the application (e.g. presentation, computations etc.).

The structure and functionality of the physical layer are beyond the scope of this thesis. In order to maintain generality of the subject layer, specific application functionality is left for implementation in the application layer. For example, the details that define the appearance of the technical information on the screen, also known as the Formatting Output Specification Instance (FOSI) (MIL-M-28001A, July 1990, Appendix B), are restricted to the application layer because different applications can have different FOSIs, yet access the same technical manual information in the database. For these

reasons the focus of the following section is placed on the remaining layer, i.e. the subject layer.

Figure 13 depicts the relationship between the different layers that compose the IETM class hierarchy.

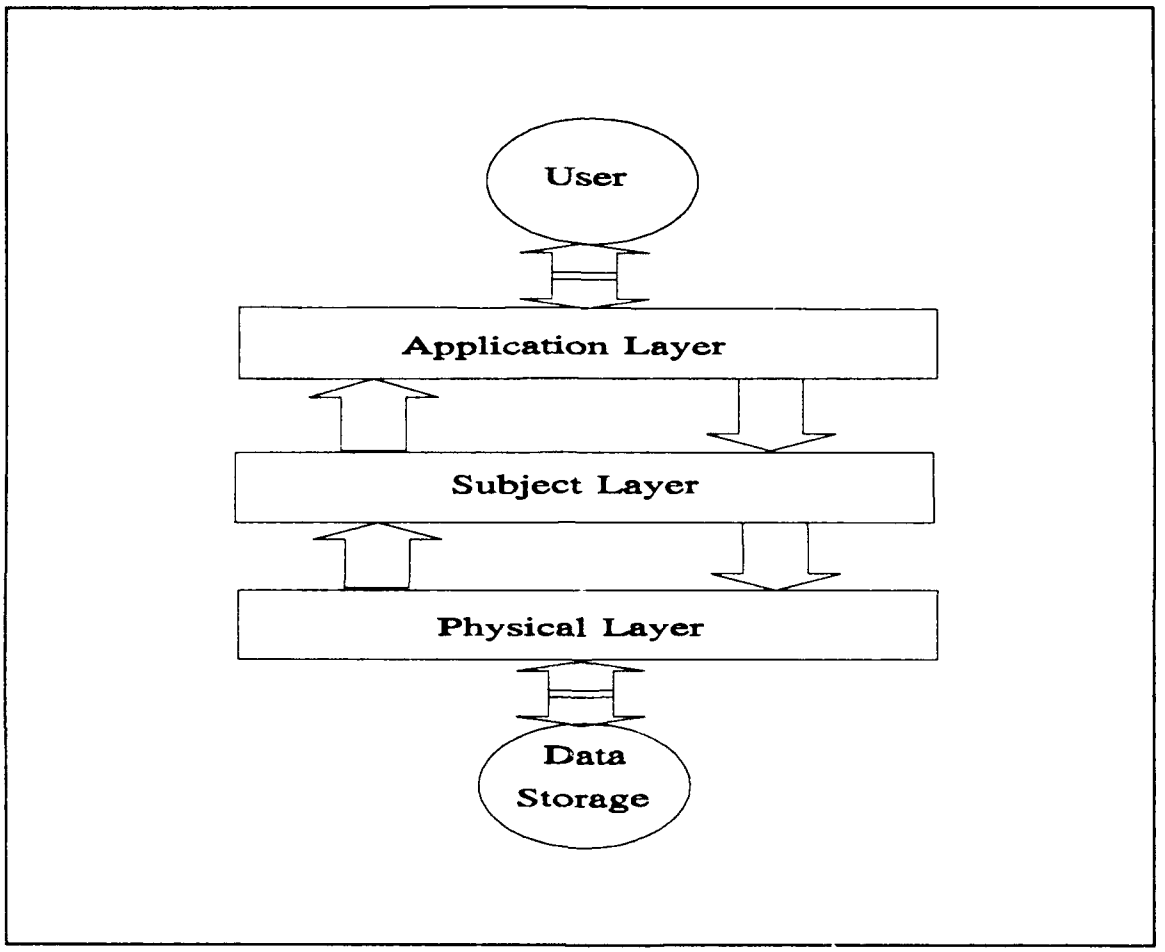

Figure 13 The IETM Object Class Hierarchy Layers

2. The IETM Subject Class Layer

The primary function of the IETM subject layer is to provide CDM functionality. This includes providing access to higher level objects as defined by the

CDM and hiding low level details from the presentation applications (or any other applications that might require CDM data access).

For the purpose of providing an example of the structure and functionality of the proposed subject layer, a closer look will be taken at the CDM structures that provide the interaction between the user and the IETM. This section of the CDM contains the "dialog" elements : dialog, dialog_alts, fillin, menu, prompt, choice and selection (MIL-D-IETMDB, April 1991, pp. A-21 through A-23). An example of a standard user interface screen that utilizes these elements is depicted in Figure 14[17]. Following the guidelines set in the previous sections, an object class hierarchy has been defined to capture the functionality of the interaction elements.

3. Functions of Interaction Classes

There are different approaches for conducting object oriented design. Following is a discussion on the functionality of the various user interaction classes, using an approach similar to the responsibility-driven design approach presented in (Wirfs-Brock and Wilkerson, 1989):

a. Dialog Class

The dialog object provides the basic interaction capabilities required by the user of the IETMDB. As can be seen in the dialog element definition:

[17] Figure 14 was adapted from (Wampler, 1991, LP-25).

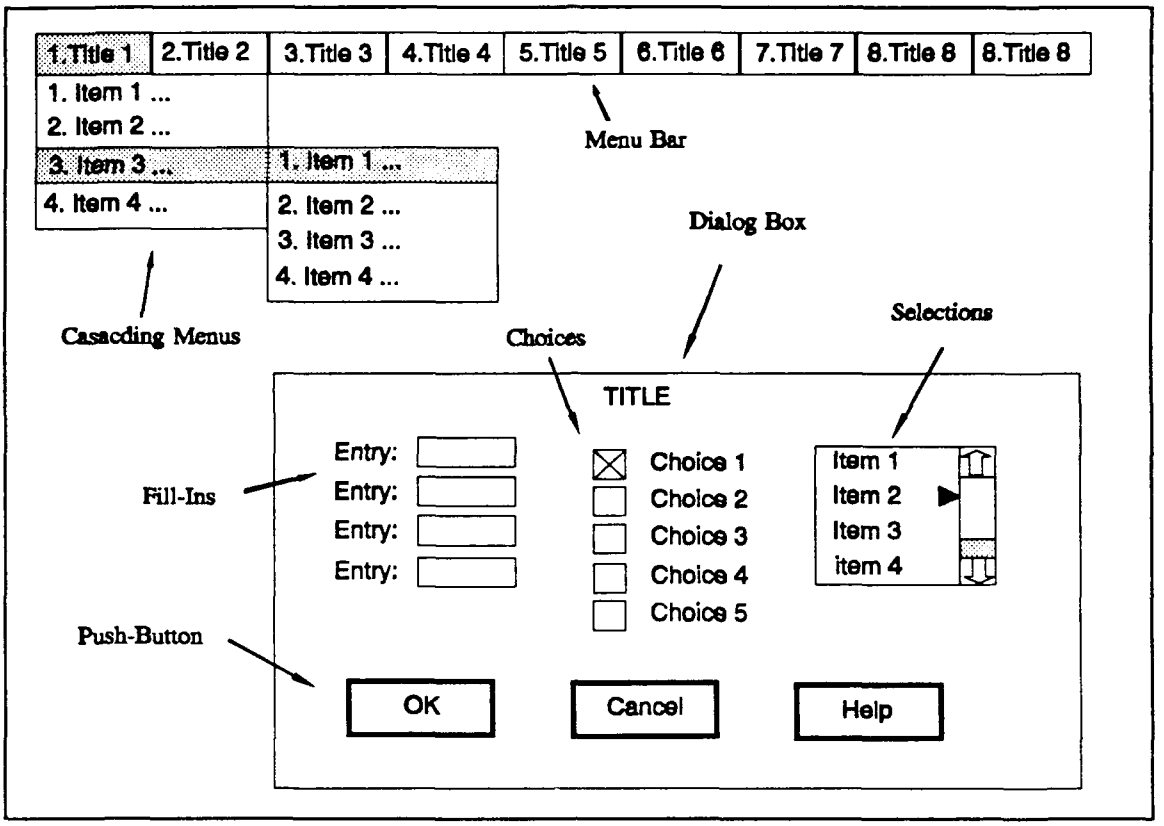

Figure 14 Objects in a Standard User Interface

```
<!ELEMENT dialog - o ( precond*, link*, ( %text;)?,
                    ( dialog_alts | dialog | fillin | menu | selection )+ )>
<!ATTLIST dialog
        %a.node;
        agent       CDATA   'human' >
```

the dialog element is an aggregation of the following subcomponents: precondition(s), link(s), an optional text title and one or more subdialog components. The basic functionality required from the dialog class object is to capture the dialog data in its instance variables through set methods and retrieve the data through get methods. All the responsibility for finer dialog functionality (behavior) is provided by the dialog subobjects (i.e. functionality is shared via aggregation). The dialog object will be therefore required

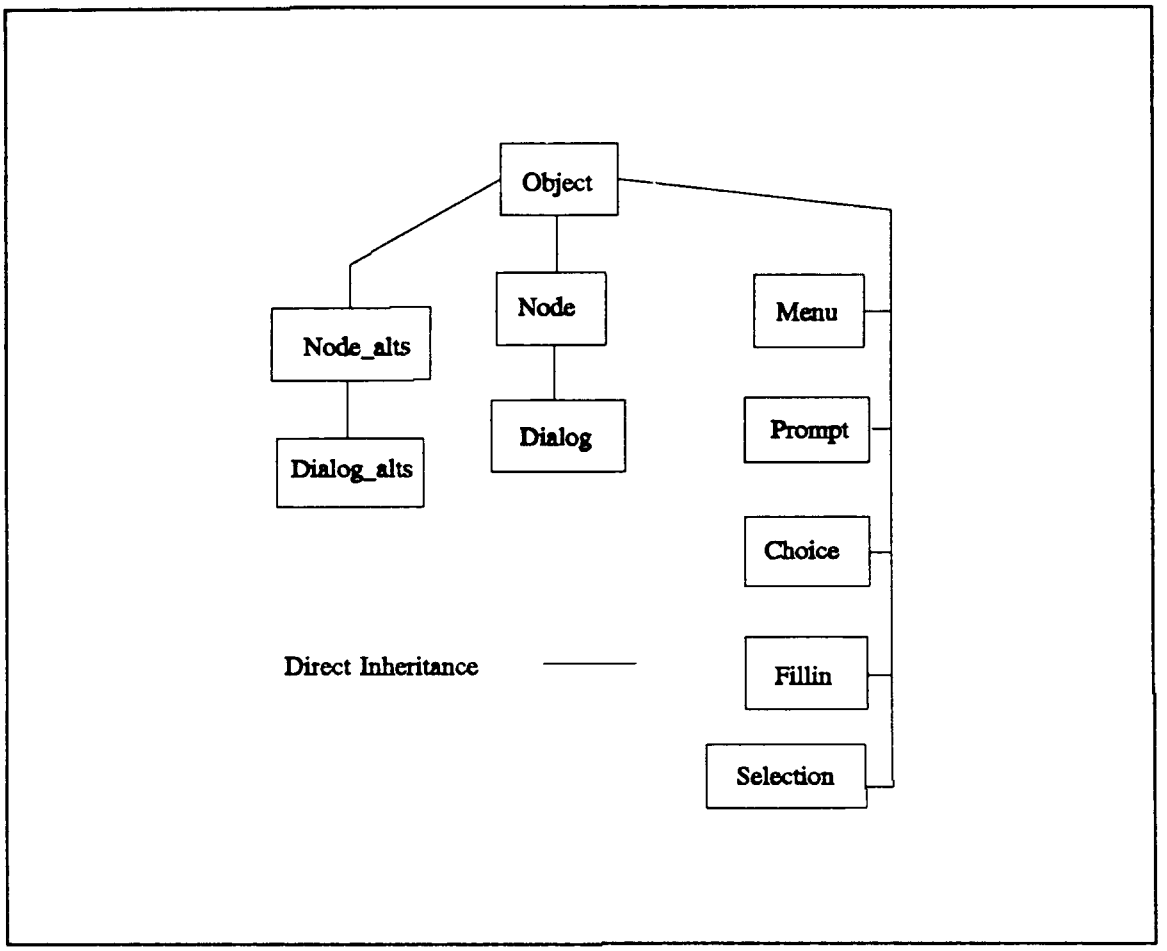

Figure 15 The Subject Layer Class Hierarchy

to capture the user input and delegate it to the appropriate component object. Since the dialog structure implies that a dialog can have multiple subdialogs, appropriate methods are needed to identify the correct subobject to which the dialog object will send the message containing the user input as one of it's arguments.

Additional functionality provided by the dialog object is embedded in its precond and link instance variables. The precondition object provides the dialog with the capability to evaluate an expression at presentation time, and according to the result allow/disallow the display of the dialog. The link instance variable provides the

capability to link the dialog to other data (such as special graphics, etc.). Therefore, the dialog object will also need to be able to test its precondition and display itself, if the precondition resolves to true, as well as activate links as a result of user request. Here again, rather than the dialog class providing the required functionality, the appropriate behavior will be provided by the classes of the receiving subobjects.

Table 10 contains the definition of the dialog class. Definitions for the other interaction classes can be constructed in a similar manner.

b. Dialog_alt Class

The dialog_alt object provides the user with the capability to make context sensitive selection of dialog objects, as well as reduce data redundancy in the database. As can be seen in the dialog_alts element definition:

```
<!ELEMENT dialog_alts  - o  ( dialog )+ >
<!ATTLIST dialog_alts
        %a.node_alts; >
```

the dialog element uses the attributes defined by the node_alt template. The dialog object will inherit these attributes from the node_alt class. The dialog_alt subcomponent is a collection (set) of dialog objects which were grouped together because they are conceptually the same dialog applied at different contextual situations[18].

[18] The filtering is performed by evaluation of the preconditions of the dialogs. The semantic rules of the node_alt template define that the dialogs should be mutually exclusive, such that the precondition of only one dialog will evaluate to true.

TABLE 10 THE DIALOG CLASS DEFINITION

Class	Dialog		
Inherits from	Node		
Inherited by	N/A		
Functionality	1. Provide access to data required for user interaction.		
Instance Variables	**Name**	**Inherited From**	**Description**
	id	Node	Item identification
	name	Node	Element name
	type	Node	Information type
	itemid	Node	Reference designator
	cdm	Node	Template type
	ref	Node	Reference to data element
	agent		Dialog counterpart type
	precond		Context dependant filtering subobject
	link		Relational links to other elements
	title		Text string to capture title
	subdialog		Either a dialog_alts, dialog, fillin, menu or selection object, or any combination of these
Class Methods			
new(self, precond, link, title, subdialog)			Create a new dialog object
Object Methods			
get_agent(self)			Return identification of dialog counterpart
get_precond(self)			Return dialog precondition
get_link(self)			Return link object to additional info
get_title(self)			Return dialog title
get_subdialog(self)			Return subdialog object
set_agent(self, agent)			Set agent instance variable
set_precond(self, precond)			Set precond instance variable
set_link(self, link)			Set link instance variable
set_title(self, title)			Set title instance variable
set_subdialog(self, subdialog)			Set subdialog instance variable

The basic functionality required from the node_alt object is the functionality required from a collection, i.e. add, delete, get_item, etc. Since the preconditions of all

objects in the collection have to be evaluated, the order of the dialogs in the collection is insignificant. The implementation of the collection will have to provide for enumeration control to assure that all objects are evaluated. Specific dialog behavior will be provided by sending messages to the selected dialog object.

c. Fillin Class

The fillin object provides the user with the capability to input data. As can be seen in the fillin element definition:

```
<!ELEMENT fillin - o ( link*, prompt, property, ( %text;)? )>
<!ATTLIST fillin
        id          ID          #IMPLIED
        ref         IDREF       #CONREF
        range       CDATA       #IMPLIED  >
```

the fillin element is constructed of a prompt which contains the question displayed to the user (this could be either a text or a graphic symbol), a property which will be the object to receive the user response, and an optional text element that defines a default value to be presented as the default fillin.

The functionality required from the fillin object is to capture the user input (and delegate it to the property subobject), display itself in the given display context (which is defined by the using application) and initialize the default value for the fillin.

d. Menu Class

The menu object provides the capability to display to the user several choices for selection. As can be seen from the menu element definition:

```
<!ELEMENT menu - o ( link*, prompt, choice+ )>
<!ATTLIST menu
        id          ID          #IMPLIED
        ref         IDREF       #CONREF
        select      ( single | multiple )    'single'  >
```

the menu is an aggregation of a prompt and a collection of available choices. The basic functionality required from the menu object is to capture the identification of the selection(s) made by the user and activate the correct choice response(s) by sending a message(s) to the correct choice subobject(s). The menu object is required to display itself in a given display context, defined by the user application.

e. Prompt Class

The prompt object provides the capability to display to the user a certain message, which can be either a text message or a graphic symbol (such that the user interprets as a message). As can be seen in the prompt element definition:

```
<!ELEMENT prompt - o ( %text; | %graphic; )>
<!ATTLIST prompt
        id          ID          #IMPLIED
        ref         IDREF       #CONREF  >
```

the prompt consists of either a text or a graphic subcomponent.

The basic functionality required from the prompt class object is to delegate a display message received by itself to the subobject that contains the prompt data.

This is a good setting to demonstrate one of the advantages of using the Object-Oriented approach by giving an example of the extensibility of OOP classes: since

the prompt object doesn't need to know the details of what object is captured as its subobject instance variable, the latter could conceptually be an audio subobject as well. All that is needed is that the subobject class (i.e. the audio class) contain an object display method (it is assumed that the intention of display is to present, which in the audio object case would be sound generation, i.e. show yourself = sound yourself). Thus the functionality of the prompt class is extended by sharing the behavior provided by the audio class object in a very straightforward manner that doesn't require excessive recoding.

f. Choice Class

The choice object is a subelement of a menu, i.e. upon it's selection a specific sequence of actions is executed. As can be seen in the choice element definition:

```
<!ELEMENT choice - o ( ( %text; | %graphic; ), ( postcond+ | %dialog ) )>
<!ATTLIST choice
        id          ID              #IMPLIED
        ref         IDREF           #CONREF
        default     ( Yes | No )    'No' >
```

the choice element contains a data subobject (which can be either a text or a graphic) and an execution subobject (which is either a postcondition or a dialog) that defines the action to be taken if the object is selected by the user.

The basic functionality required from the choice object is to display its data subobject and to execute the appropriate action associated with the choice. The former is done by sending a display message to the data subobject (both text and graphic objects

have display methods), whereas the latter is done by sending an execute message to the execution subobject.

g. Selection Class

The selection object provides the user with the capability to make a selection from a given picture, text string or table. As can be seen in the selection element definition:

```
<!ELEMENT selection - o  (  ( link, ( postcond+ | %dialog; ) )+,
                            ( text | table | graphic ) ) >
<!ATTLIST selection
          id          ID          #IMPLIED
          ref         IDREF       #CONREF  >
```

the selection element contains three subcomponents - the link, execution and data subcomponents. The semantics defined by this structure are that each selection element defines pairings of links with selectable data elements (which are either text, table or graphic elements). Each link must have a link-end specifying the selectable (data) item. Each link is paired with an execution subelement which defines the action to take once the selection is chosen.

The functionality required from selection class objects is to display the selection object (by sending a display message to the data subobject) and to execute the appropriate action defined by the selection by sending an execute message to the execution subobject.

4. The IETMDB System Environment

The IETMDB is not a stand-alone system, but rather a part of a system environment that facilitates storage in, and retrieval from, the IETMDB.

The primary component of this system environment is the IETMDB Loading Program. This program accepts two kinds of input: the DTD which describes the technical information CDM and the Technical Information Document (TID). The loading program checks that the TID syntax is in accordance with the grammar defined by the DTD, and creates the appropriate objects in the database. This means that the loading program will have the knowledge of which type of aggregations to create (i.e collections, dictionaries etc.), and that the correct objects will be created in the database, as the parsing of the TID commences.

Figure 16 contains a depiction of the IETMDB system environment.

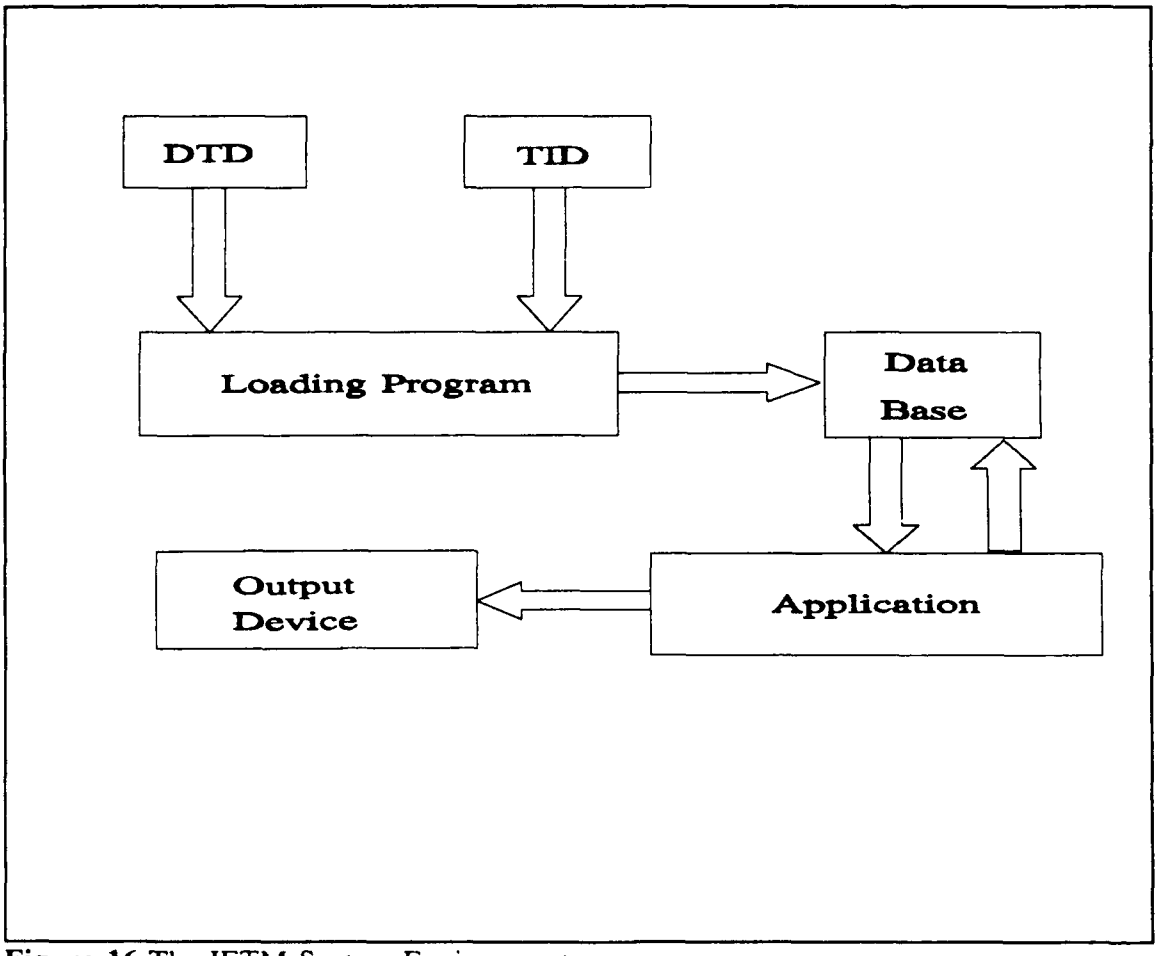

Figure 16 The IETM System Environment

VIII. BENEFITS OF USING OODB FOR IETMDB IMPLEMENTATION

A. INTRODUCTION

The previous chapter was focused on presenting the proposed solution, namely demonstrating how an IETMDB can be designed by utilizing object-oriented design concepts and methodology. The purpose of this chapter is to present the pros and cons of using Object-Oriented Data Base (OODB) technology for implementation of IETMDB. The first section provides background on some of the core object-oriented concepts and capabilities, and compares them to those provided by Relational Data Base (RDB) technology. This comparison is mainly for the purpose of emphasizing the unique features of OODBMS. The second section contains a description of some OODB architectural issues that are closely related to application performance. The final section discusses the unique features of OODB in the context of implementing an IETMDB. The discussion is targeted at obtaining a better understanding of the extent that an IETMDB implementation can exploit OODB capabilities and features.

B. CORE OBJECT-ORIENTED CONCEPTS

Although a prescribed standard for OODB systems does not exist yet, some concepts of OOP can generally be regarded as core object-oriented concepts (Kim, 1991, pp. 22-23). These concepts are: encapsulation, classification, inheritance, aggregation and

polymorphism (Kim and Lochovsky, 1989, Part 1). Each of these concepts is hereby briefly described.

1. Encapsulation

Objects encapsulate both data and program code. In OOP terminology these are known as the object state (i.e. the values of the object attributes or instance variables) and the object behavior (i.e. the methods that are recognized by the object, and that cause a change in the object state).

The RDB technology counterpart of an object is a tuple. In most RDBMS implementations a tuple can only capture data, and not behavior. However, some recent efforts have been conducted in the area of Extended Relational Data Bases (ERDB) in order to incorporate references to procedures as attributes of tuples and allow complex data types[19].

2. Classification

Classes are groupings of objects that share the same set of attributes and methods. Hence, an object is an instance of a class.

In the RDB, a table (relation) groups together tuples that share the same attributes. As stated earlier, in most implementations these attributes capture data values alone, and not behavior. In the case of ERDB the value of a behavioral attribute need

---

[19] Postgres (an extension of the Ingres RDB) is an example of ERDB. A shared complex object in Postgres is represented by a field that contains a sequence of commands to retrieve data from other relations that represent the subobjects. Code replication can be avoided by storing the procedure in a separate relation (i.e. normalization is required) and by passing the object to the procedure as an argument (UCB/ERL, M86/85, 1987, pp. 6-8).

not be the same for all tuples, hence behavior is not generally shared amongst all tuples in a relation[20].

3. Inheritance

Classes are organized in a class hierarchy which is a rooted, directed acyclic graph, such that classes inherit all the attributes and methods of both direct and indirect ancestor classes. A class is said to be a specialization of classes it inherits from, and conversely a generalization of classes that inherit from it. Single inheritance implies that the class can have only one direct ancestor, whereas multiple inheritance implies that the class can have multiple direct ancestors.

In the RDB, the notion of inheritance does not exist. Furthermore, in some instances, in order to share data between two relations it is necessary to create a third relation.

4. Aggregation

The domain of an attribute of a class can be any class, hence the value of an attribute of an object may also be an object or even a set of objects. The attributes of an object form an aggregation of other objects, that can be either simple objects (e.g. integer, string etc.) or complex objects (i.e. that their attributes form aggregations as well). Aggregation is another form for sharing data and behavior, which in some applications can be defined dynamically (dynamic binding). The aggregation relationships define a

[20] Sharing can be simulated by having the same attribute value for all tuples, but this implies redundancy.

directed graph of classes, that may be cyclic. This graph is sometimes called the Aggregation Hierarchy, to contrast it from the Inheritance Hierarchy described earlier.

In the RDB, the domain of an attribute is restricted to primitive data types. The attribute can have only a single value (and not a set of values) and the data type of the attribute can not change dynamically.

5. Polymorphism

The term polymorphism describes a situation in which objects of different classes can invoke the same method (i.e. the receiver of the message can take many forms). This feature is especially important in the context of extensibility and reuse of software.

Polymorphism is a feature unique to OOP; however, the RDB have no characteristic with similar capabilities.

C. OTHER OBJECT-ORIENTED CHARACTERISTICS

Beside the core object-oriented concepts mentioned previously, there are other characteristics of OOP that should be mentioned in the context of this study. These are the concepts of the object identifier and the consistent data model:

1. Object Identifier

In the RDB, the key (identifier) of a tuple is the values of a set of attributes. These attributes represent real world data, which if altered cause the definition of a new tuple. In OODB each object has a unique identifier, thus the attributes of the object can change without the object losing it's identity (Loomis, 1990, p. 81). The representation

of complex objects can be conducted uniformly by the value of the attribute being set to an object identifier (or a set of identifiers) of instances of the attribute domain (Kim, 1991, p. 24).

2. Consistent Data Model

In the RDB environment, the data models used for the application programming language and the database manipulation language are often different, thus leading to what is known as the "impedance mismatch" (Copeland and Maier, 1984). By using OOP, the same data model can be used by both the application and the database interface, hence providing the basis for eliminating the need for a separate database programming language and maintaining a consistent data model throughout the software development process (Loomis, 1990, pp. 81-82).

D. OODB ARCHITECTURAL ISSUES

Many architectural issues can be discussed in the domain of database design. Amongst these are storage management, indexing techniques, concurrency control, optimization, version control, and integrity. The purpose of the following discussion into the domain of architectural issues is to investigate if a relationship exists between characteristics of the IETM application (as defined by its functional and performance requirements) and a preferred design of the underlying OODB which is used to support it. In other words, are there architectural characteristics of OODB that would render a certain OODBMS implementation more suitable than another, for the purpose of implementing IETMBD? The following subsections will provide some background,

whereas the actual evaluation in the context of IETMDB will be deferred to the following section.

1. Clustering of Data

Clustering is a technique used to store a set of related objects close together in secondary storage, such that the cost of Input and Output (I/O) operations required to retrieve them is minimized. In the RDB, clustering is more straightforward, since tuples of the same relation can be clustered on the same segment of disk pages, thus maintaining the integrity of the logical entity (i.e. the relation) in physical storage. The fact that complex objects can be stored in OODB gives room for multiple options for clustering data, some of which are detailed in (Kim, 1990, p. 32).

Since every clustering option is optimal only to a certain access scenario (i.e. dependant on the type and frequency of object accesses defined by the nature of the application), it is important to analyze the user application access scenario to determine if a certain clustering strategy would provide higher performance from the user perspective.

2. Queries

Because the domain of an object attribute can be any class, a query in an OODB can result in considerable data retrieval because the query is formulated against the nested definition of a class (Kim, 1990, p. 33). Although it has been shown that object-oriented queries can be evaluated in a similar manner to relational queries, some query optimization problems have yet to be solved (Kim, 1990, p. 33)(Kim, 1989).

It is therefore necessary to evaluate the nature of expected queries in a given application because the required operations to preform the query (e.g. selections, joins, etc.) places performance demands on the OODBMS.

3. Authorization

In the RDB, the smallest unit of authorization is a relation or an attribute of a relation. In OODB the smallest unit of authorization should logically be an object, but that might be potentially expensive, since the database will have to maintain authorization triples (i.e. object i.d., authorization type, authorized user) on each object rather than on each class (Kim, 1990, p. 37). Besides the common authorization types (create, read, update) it is necessary to define new authorization types for creating a subclass, and executing and changing methods (Kim, 1990, p. 37). Thus, when evaluating an OODB for a certain implementation, it is also necessary to consider the authorization requirements of the application, and potential use of the new authorization types suggested earlier.

E. UTILIZING OODB CAPABILITIES IN AN IETMDB IMPLEMENTATION

As mentioned earlier, no structural requirements are imposed by MIL-D-IETMDB on the actual DBMS methodology used to implement IETMDB (i.e. the implementation methodology can be either relational or object-oriented) (MIL-D-IETMDB, April 1991, p. 3). Having presented in the previous sections some of the unique features of OODBMS, and the merits of OODB in comparison with RDB, it is appropriate at this stage to attempt to establish the thesis that OODB capabilities can be utilized when

implementing an IETMDB, thereby resulting in a smaller development effort than the one that would have been required if RDB technology had been used. This will be done by evaluating OODB unique features and architectural characteristics in the context of implementing an IETMDB.

1. Using Encapsulation

The fact that an object encapsulates both the data and the behavior associated with the data, relieves the application program from the responsibility of redefining this behavior upon every access to the data, thereby enhancing sharing. In an OODB implementation of an IETMDB, both the prompt and the choice objects[21] can have subcomponents which are graphic objects. Because the graphic object encapsulates the data (i.e. the image) and the behavior associated with it (e.g. display the image), both the choice and the prompt objects are relieved from the need to define how to display their graphic subcomponents. Both objects share the behavior defined by the graphic object, and conversely the graphic object behavior is defined only once. As mentioned earlier this sharing capability is not provided by the RDB. It can be implemented in the ERDB by creating a graphic relation with the procedural attribute having redundant values (e.g. the display procedure), or in a normalized version by creating one relation to capture the graphic data and another relation to capture the behavior. Both ERDB solutions are much more complex, and therefore more difficult to design, than the proposed OODB alternative.

---

[21] See details on the prompt and choice objects in Chapter VII, Section D, Subsection 3.

2. Using Classes and Inheritance

For each element in the CDM a corresponding class is defined in the OODB[22]. Inheritance was used to implement the reusability characteristics of generic layer templates, by defining them as abstract classes from which other generic layer elements (classes) and content specific elements (classes) inherit[23].

As mentioned earlier the concept of inheritance does not exist in either the RDB or the ERDB. A solution using RDB technology will require a more complex schema to support the implementation of the generic layer templates and their reuse by content specific elements.

3. Using Aggregation

Data typing in OODB is much less constrained than in the RDB (Loomis, 1990, p. 81). Furthermore, OODB allows complex objects to be modeled as collections of objects of primitive data types and arbitrarily complex objects that in turn consist of these complex objects (Kim, 1991, p. 24).

In the IETMDB, the dialog object is such a complex object[24]. The subcomponents of the dialog object form an aggregation of primitive data types (e.g.

---

[22] See discussion on mapping of CDM structures to object classes in Chapter VII, Section C, Subsection 1.

[23] See discussion on the mapping of CDM structures to object classes in Chapter VII, Section C, Subsection 1.

[24] See details on the dialog object in Chapter VII, Section D, Subsection 3.

text) as well as complex data types (e.g. dialog, dialog_alts). The subcomponents can have multiple occurrences, which means that the dialog object can be arbitrarily complex.

The functionality required from the dialog object is to support the entire interactive process, which is a task requiring multiple data types and numerous methods. Yet, these are not provided directly by the dialog object itself, but rather through a carefully defined aggregation of subcomponents, each carrying its own data and behavior (i.e. functionality) to be shared by the top level dialog object. Although the aggregation hierarchy can become very complex (especially because the underlying graph can be cyclic), the high level abstract data type of a dialog can still be supported. As mentioned earlier, this level of abstract data typing is not available in RDB technology.

The significance of being able to define arbitrarily complex objects to implement the user interaction with the IETM system is that the human interface engineer can exercise a high degree of freedom in the design of the interface: no restrictions on menu hierarchy depth, freedom to switch interaction objects (i.e. fill-in, selection list, etc.) in any level and between levels. All this can be achieved without imposing additional complexity on the schema design, which would have been the case if RDB technology had been used.

4. Using Polymorphism

Polymorphism provides the capability for enhancing software reusability (Nierstrasz, 1989, p. 10) and improving code extensibility (Wu, 1990). An example for polymorphism in an IETM application is that both a text object and a graphic object can

be potential receivers of a display message. This can be extended even further to include an audio object[25].

5. Clustering Strategy for IETMDB

Kim (1990, p. 32) lists four options for clustering objects on secondary storage:

1. Clustering all objects belonging to the same class in the same segment of disk pages (i.e. the inheritance and aggregation hierarchies are disregarded).
2. Clustering of objects belonging to a class hierarchy rooted at a user specified class (i.e. candidates for clustering are determined by the inheritance hierarchy).
3. Cluster together objects and other objects that they recursively reference (i.e. candidates for clustering are determined by the aggregation hierarchy).
4. Cluster together classes on a class hierarchy (rooted at a user specified class) and a subset of class attribute graphs rooted at these classes (i.e. combination of option no. 2 and no. 3).

Clustering of objects is optimal for a certain object access scenario which is application dependant. The IETM usage scenario is expected to be similar to the scenario of the regular maintenance routine preformed by a technician using a hard-copy manual: there is a beginning task (root task) which has a sequential list of subtasks or steps. Occasionally the technician jumps to another section of the manual to perform a perquisite task (e.g. safety inspection before disassembly of a part) or reference a detailed

[25] See discussion on extensibility in the context of the audio object in Chapter VII, Section D, Subsection 3.e.

diagram. He then returns to the main list to preform the next step (which could be the root of a new task).

Clustering option no. 1 does not seem appropriate for IETM implementation because the IETM access scenario does not require a sequential scan of all objects in a certain class. Option no. 2 could be used, but would not render the expected performance improvements since the IETMDB inheritance hierarchy is very shallow. Option no. 3 seems the most appropriate for IETMDB implementation because of the nature of the CDM element subcomponents: preconditions, links, etc. Preconditions have to be evaluated in order to determine if the object can be displayed whereas the links to other objects should be available such that no search is required if the user elects to traverse a link to another piece of data.

The issue of defining a preferred clustering strategy for IETMDB implementation requires further research and collection of empirical data.

6. Queries in an IETMDB

The IETM is not a query oriented system. The retrieval of data is conducted according to the hierarchical navigation defined by the user input. All navigation options are predetermined and the next step/branch is preformed by "pointing and clicking". Furthermore, some of the hardware devices contain no keyboard in order to support the simplicity of IETM usage (i.e. by restricting the allowable user input types).

If this philosophy of an IETM with no queries is maintained, than no special evaluation of the nature of expected queries is required. However, in the event that the IETM interface specification is altered in the future to allow search and query mode (in

addition to navigation mode), then a careful study of expected query performance should be conducted. It is important to note that an OODB has built in provisions for executing "class searches" similar to the table searches conducted in RDB. The fact that each object has a unique identifier, and that each object is an instance of a specific class provides us with the capability to access all instances of a given class, even though the objects might not be stored together in secondary storage.

7. Authorization Control in an IETMDB

The creator of an IETM (i.e. the author of the technical manual) and the field user (i.e. the technician) are two distinct entities. The authorization to create a new subclass falls only to the IETM development domain, and could be used to restrict dissemination of data from existing classes to newly defined classes. In the user domain, the restriction of access to certain data is performed by means of context dependant filtering (MIL-D-IETMBD, 1991, p. A-3). Therefore, placing additional restrictions in the form of authorization to perform methods is not really required.

The details of authorization control in an IETM authoring environment are left for future research.

8. Using a Consistent Data Model

As pointed out earlier, OOP facilitates using the same data model for both the application and the database interface, hence eliminating the need for a separate database programming language. This reduces the total effort required for the system software development process and contributes to cost reductions in the software maintenance phase,

as well. Although this benefit is applicable to any application domain, and not only to the IETM application, it is appropriate to end the discussion on the benefits of using OODB for IETMDB implementation by re-emphasizing this important feature. New technologies need to be applied in order to decrease the magnitude of software maintenance costs, and OODB technology has good potential in helping the software industry move forward to achieve this goal.

IX. CONCLUSIONS

The main goal of this thesis was to gain better understanding of the potential use of OODB technology for implementation of CALS systems. The IETM system served as an excellent candidate for this purpose, being a system which is currently at the front-edge of CALS development efforts of all three services: Navy, Air-Force and Army.

Based on the MIL-D-IETMDB specification, an OODB was designed for implementing the IETMDB. The benefits of using an OODB approach for the implementation of IETMDB were demonstrated, and discussed in comparison to the alternative of implementing the IETMDB by means of the well established RDB approach. The conclusion, in this respect, is that not only are none of the RDB capabilities given up by adopting the OODB approach, but on the contrary, there is much to be gained: a unified data model and a simple schema contribute tremendously to the overall reduction of the effort required for the development and maintenance of software systems in general, and IETM systems in particular.

OODB is still a young technology, and not all of the issues concerning it are yet resolved. There are many architectural aspects of OODB that might have an impact on IETMDB performance, some of which were touched upon earlier in the thesis. There still remains a large area for further research in the domain of the architectural characteristics of OODB, before one will be able to empirically establish that it is a superior approach over the RDB approach, for a given application domain. This is the main reason that this

study is qualitative in nature: theory and practice have to advance even more before quantitative tools can be developed and put into use to assist in reaching a decision as to the preferred DBMS approach for a given application.

Another problem that is not addressed by this new technology is the problem of the enormous install-bases of data that exist in earlier forms of DBMSs. Clearly, it seems unlikely that an organization would invest in converting its databases to a new form of technology, whenever the latter becomes available. The cost will be far to great. Thus alternative approaches should be researched, as suggested by Hsiao in his paper on Federated Databases and Systems (1992). This, and other topics for future research are detailed in the final chapter.

X. FUTURE RESEARCH

Because of the limited time frame allocated to the thesis research detailed in this document, it was impossible to study in depth some of the topics that were defined initially, and others that were not planned but surfaced during the study. These topics are hereby listed, together with some of the questions that remain open for future research:

1. Implementation of an object-oriented IETMDB. All IETM implementations presented in the CALS 1991 Exposition utilized RDB for IETMDB implementations. Significant knowledge will be obtained by the actual implementation of an object-oriented IETMDB, as advocated in this thesis.

2. A preferred clustering strategy for IETMDB. The strategy recommended in this thesis needs to be evaluated in a laboratory environment, and the results compared to those obtained by utilizing other strategies.

3. Authorization control in an IETMDB. Details of authorization control in the IETM authoring environment need to be studied more closely. It is not clear if the mechanisms built into the data model are sufficient.

4. Using the Federated Database concept in an IETM systems environment. During the initial testing of the CDM, data from the gun system of both the F-16 and the F-15 aircraft (which are equipped with the same gun) was input into the database. This test revealed inconsistencies in data of the same (weapon) subsystem across different (vehicle) platforms. Thus the problem of putting existing technical data into IETMDBs is eminently clear, and it gets even worse when different data models are used for data representation. Can a Federated Database approach serve as a practical solution for this problem? What DBMS approach should be used to concur the mountains of existing technical data, given that a requirement for IETMDB will be defined not only for future weapon systems (i.e. as they are developed) but also to the hundreds of weapon systems that exist today?

LIST OF REFERENCES

Air Force Human Resources Laboratory, AFHRL-TP-90-10, *Content Data Model: Technical Summary,* by Earl M. and Gunning D., May 1990.

Andrews T. and Harris C., *Combining Language and Database Advances in an Object-Oriented Development Environment,* OOPSLA '87 Proceedings, October 1987; Special Issue of SIGPLAN Notices, Vol. 22 No. 12, December 1987, pp. 430-440.

Bracket M. H., *Developing Data Structured Databases,* Prentice-Hall Inc., 1987.

Caporlette B. K., *Integrated Maintenance Information System (IMIS), Content Data Model (CDM),* Proceedings of the 4th Annual CALS Progress Review, November 1991, pp. AP72-AP80.

Copeland G. and Maier D., *Making Smalltalk a Database System,* Proceedings of the ACM SIGMOD, June 1984, pp. 316-325.

Cox B. J., *Object-Oriented Programming: An Evolutionary Approach,* Addison-Wesley Publishing Co., 1986.

David Taylor Research Center, DTRC-89/007, *The Electronic Delivery of Automated Technical Information for Logistics Support of Navy Weapon Systems: Potential, System Description and Status*, by Rainey S. C., Fuller J. J. and Jorgensen E. L., February 1989.

David Taylor Research Center, DTRC-90/026, *Proposed Draft Military Handbook for Preparation of View Packages in Support of Interactive Electronic Technical Manuals (IETMs),* by Rainey S. C., Jorgensen E. L. and Fuller J. J., July 1990.

Department of Defense, *Computer-aided Acquisition and Logistic Support,* 1989.

Department of Defense, Directive 5000.2.

Department of Defense, MIL-M-28001A, *Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange of Text,* July 1990.

Department of Defense, Draft MIL-STD-1388-2B, *DoD Requirements for a Logistic Support Analysis Record,* January 1991.

Department of the Navy, *CALS Architecture/Implementation Plan,* 1991.

Department of the Navy, *CALS Strategic Plan*, 1988.

Hsiao D. K., *Federated Databases and Systems*, to appear in the International Journal on Very Large Databases (VLDB Journal), Vol. 1, No. 1, March 1992.

International Standards Organization, ISO/IEC CD 10744, *Information Technology - Hypermedia/Time-based Structuring Language (HyTime)*, April 1991.

Kim W., *A Model of Queries for Object-Oriented Databases*, Proceedings of the International Conference of Large Data Bases, August 1989.

Kim W. and Lochovsky F. H., *Object-Oriented Concepts, Databases and Applications*, Addison-Wesley Publishing Company, 1989.

Kim W., *Architectural Issues in Object-Oriented Databases*, Journal of Object-Oriented Programming, March-April 1990.

Kim W., *Object-Oriented Database Systems: Strengths and Weaknesses*, Journal of Object-Oriented Programming, July-August 1991.

Loomis M. E. S., *ODBMS vs. Relational*, Journal of Object-Oriented Programming, July-August 1990.

National Security Indastrial Association (NSIA), *"CALS: Making It Happen"*, CALS Expo '91 Conference & Exposition Guide, November 1990.

Naval Po raduate School, Report NPS52-90-025, *Object-Oriented Database Managment Systems*, by Nelson M. L., May 1990.

NAVINSGEN, *Review of Navy Technical Manual Program*, 1984.

Nelson M., Moshell J. M. and Orooji A., *A Relational Object-Oriented Management System*, 9th Annual International Phoenix Conference on Computers and Communications (IPCCC '90) Proceedings, March 1990, pp. 319-323.

Nierstrasz O., *A Survey of Object-Oriented Concepts*, in *Object-Oriented Concepts, Databases, and Applications*, edited by Kim W. and Lochovsky F. H., Addison-Wesely Publishing Company, 1989, pp. 3-22.

Premerlani J. et al., *An Object-Oriented Relational Database*, Communications of the ACM, Vol 33 No. 11, November 1990.

Tri-Service Working Group for Interactive Electronic Technical Manuals, *Draft MIL-D-IETMDB, Data Base, revisable: Interactive Electronic Technical Manuals, for the Support of,* Fuller J. J. et al., April 1991.

Ullman J. D., *Principles of Database Systems*, Computer Science Press, 1982.

University of California Berkeley, Electronics Research Laboratory, Memorandum No. UCB/ERL M86/85, *The Postgress Papers*, edited by Stonebraker M. and Rowe L. A., June 1987.

Wampler J., *A Common User Interface for Maintenance Information Systems*, Proceedings of the 4th Annual CALS Progress Review, November 1991, pp. LP22-LP23.

Wirfs-Brock R. and Wilkerson B., *Object-Oriented Design: A Responsibility-Driven Approach*, Object-Oriented Programming Systems, Languages and Applications 1989 Conference Proceedings (OOPSLA '89), pp. 71-75.

Wu C. T., *Benifits of Abstract Superclass*, Journal of Object-Oriented Programming, February 1991, pp. 57-62.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Chairman, Computer Science Dept., Code CS Computer Science Department Naval Postgraduate School Monterey, California 93943-5002	1
4.	Chief of Naval Research 800 N. Quincy Street Arlington, Virginia 22217-5000	1
5.	Curriculm Officer Computer Technology Program, Code 37 Naval Postgraduate School Monterey, California 93943-5002	1
6.	Professor C. Thomas Wu, Code CS/Wq Computer Science Department Naval Postgraduate School Monterey, California 93943-5002	1
7.	CDR B. B. Giannotti NROTC Unit RAS104 University of Texas Austin, Texas 78712-1184	1
8.	Mr. Shawn P. Magill Naval Air Systems Command (AIR-41144) Washington, District of Columbia 20361-4110	1

9. CDR Stephen M. Carr, SC, USN — 1
Naval Air Systems Command (AIR-4114A)
Washington, District of Columbia 20361-4110

10. Mr. Joseph Garner — 1
CALS Technology Integration Lab, Code 185
David Taylor Research Center
Bethesda, Maryland 20084-5000

11. Israel Air Force Attache — 1
Embassy of Israel
ATTN: Commander, Systems Divison
3514 International Dr., N.W.
Washington, District of Columbia 20008

12. Major Evyatar Chelouche — 2
P.O. Box 67
Reut, 71908
Israel